Image Multi-Distortion Estimation

André L. Caron Pierre-Marc Jodoin
MOIVRE Research Center
Université de Sherbrooke
Sherbrooke, Canada

***Abstract*— We present a method for estimating the amount of noise and blur in a distorted image. Our method is based on the MS-SSIM framework which, although designed to measure image quality, is used to estimate the amount of blur and noise in a degraded image given a reference image. We show that there exists a bijective mapping between the 2D noise/blur space and the 3D MS-SSIM space which allows to recover distortion parameters. That mapping allows to formulate the multi-distortion estimation problem as a classical optimization problem. Various search strategies such as Newton, Simplex, NewUOA, and brute force search are presented and rigorously compared. We also show that a bicubic patch can be used to approximate the bijective mapping between the noise/blur space and the 3D MS-SSIM space. Interestingly, the use of such a patch reduces the processing time by a factor of 40 without significantly reducing precision. Based on quantitative results, we show that the amount of different types of blur and noise in a distorted image can be recovered with an accuracy of roughly 2% and 8%, respectively. Our methods are compared to four state-of-the-art noise and blur estimation techniques.**

I. Introduction

In the last decade, various quality metrics have been designed to measure the visual distance between two images. Examples of such metrics are Multi-Scale Structural SIMilarity (MS-SSIM) [1], Wavelet Structural Similarity (WSSI) [2], Visual Signal-to-Noise Ratio (VSNR) [3], and Visual Information Fidelity (VIF) [4] indices, to name a few. These metrics are designed to give a score between a reference image f and a distorted version of it g. From that score, one can conclude that the quality of image g is from excellent to very bad.

Unfortunately, those quality metrics share a common limitation as they cannot identify the kind (nor the amount) of degradation that has been applied on f to obtain g. Furthermore, given a visual score, those methods cannot determine if one or more degradations has been applied to the reference image. In other words, adding noise and/or blur to an image reduces its visual score, but none of these metrics can identify which degradation and how much of it has been applied. Although some methods have been proposed to estimate the amount of one single distortion [5]–[8], none is capable of estimating simultaneously the amount of multiple distortions. At most, multiple distortions are estimated independently [8].

In this paper, we introduce a new method to estimate the amount of multiple distortions in an image g given a reference image f. More specifically, we present how the noise (α) and blur (β) distortion parameters can be recovered from an image pair (f, g), two distortions pivotal for many denoising and deconvolution techniques [9]. Our method is based on the MS-SSIM distortion factors which has been designed to measure image quality. As will be shown below, a slight extension of MS-SSIM allows to recover the noise and blur distortion parameters. This extension is based on the MS-SSIM distortion factors accounting for luminance distortion (ld), contrast distortion (cd) and structure distortion (sd) between f and g. These factors were originally used to compute the MS-SSIM quality metric only [1] [1]. In this paper, ld, cd and sd are used for a different purpose : they define the axes of a 3D space which we call the *MS-SSIM space*. In that space, each image pair (f, g) is mapped to a 3D point (ld, cd, sd). Because g is derived from a reference image f by applying distortion based on two parameters (α, β), our method focuses on the relation there is between a pair (α, β) and a 3D point in the MS-SSIM space. As will be shown below, due to a bijective transformation between the (α, β) space and the (ld, cd, sd) MS-SSIM space, one can formalize the problem of multi-distortion estimation as a classical optimization problem, *i.e.* recover the (α, β) pair whose 3D point (ld, cd, sd) is the closest to (f, g)'s 3D point. Since computing (ld, cd, sd) from a given (α, β) pair is computationally expensive, we propose a patch-based transfer function that maps any 2D point (α, β) to its 3D MS-SSIM counterpart with only 4 matrix multiplications. Different types of blur and noise can be recovered with our method including Gaussian and average blur as well as Gaussian and salt-and-pepper noise.

One application for such a method is the calibration of imaging systems whose distortion process is not subject to change in time like surveillance cameras or telescopes. Given a known reference image f and a picture of it taken by the system (here g), our method can recover the amount of blur (read point spread function) and noise introduced by the system. Such information is crucial to most restoration techniques [10], [11]. Our method could also be used in a similar fashion with lossy compression techniques.

The rest of the paper is organized as follows. We start off with an overview of the distortion estimation methods in section III. The MS-SSIM distortion factors are then introduced in more details in section IV. The bijective transformation there is between a distortion pair (α, β) and the MS-SSIM space is then explained in section V. The optimization problem (together with various optimization methods) is then formalized

This research was supported by the NSERC Discovery Grant 371951.

[1] The MS-SSIM quality metric is obtained by multiplying the three factors ld, cd and sd.

in section VI and VII. Section IX and X show results and draw conclusion.

II. Our Contributions

This paper proposes a unique MS-SSIM-based multi-distortion estimation procedure. The novelty of our method is sixfold.

1) The noise and blur parameter estimation is formalized as a search procedure embedded within the MS-SSIM framework.
2) The distortion parameters (α, β) are estimated simultaneously.
3) The simplest optimization procedure (the brute force search) is trivial to implement and returns surprisingly accurate results (approximately 2% error for blur and 8% for noise).
4) The patch-based transfer function linking the 2D noise/blur space to the 3D MS-SSIM space reduces the processing times up to 40 times without significantly reducing the results' accuracy.
5) Our method is trivially adapted to different types of blur and noise.
6) A simple extension of our method can be used to blindly estimate distortion.

III. Previous Work

When addressing the noise or the blur estimation problem, one has to consider image denoising and debluring methods. The reason being that distortion estimation has always been tightly bound to restoration techniques. Although joint denoising and debluring techniques have been proposed [12], these problems are usually independently studied or, at most, solved separately [8]. This section presents an overview of existing noise estimation and the blur estimation methods.

a) Noise estimation: The noise degradation model usually accounted for by denoising techniques is the following [13]

$$g(x,y) = f(x,y) + \mathcal{N}_\alpha(x,y),$$

where $f(x,y)$ is the intensity of the original image at pixel (x,y), $g(x,y)$ is the intensity of the degraded image at pixel (x,y) and $\mathcal{N}_\alpha(x,y)$ is additive noise with α parameter. Noise is often assumed to be Gaussian with zero mean and a variance α^2 constant throughout the image. In this case, variance α^2 is the "α" parameter of noise we mentioned in section I.

One straightforward way of estimating noise is by separating the original image f from g. In this way, noise is estimated based on the residual image void of f. Since f is usually unknown, it thus needs to be estimated. Recovering noise by first estimating f from g is a typical blind noise-estimation problem. For example, Shin et al. [13] estimate f by simply filtering g with a low-pass Gaussian filter. More complex filters have been proposed such as the thin-plate smoothing spline model [6], [7], [14]. Standard thin-plate spline methods estimates f by minimizing a cost function made of a data term and a smoothness term [14]. The data term contains an estimated version of f made of overlapping patches centered at each pixel. Another way of removing f from g is by filtering g with an high-pass filter. In [15], Rank *et al.* use a cascade of two l-D high-pass filters.

Of course, separating two images is a fundamentally difficult task and naive implementations can only lead to poor results. That is why some methods estimate noise based on g only. One strategy is to estimate noise in uniform regions void of edges [16], [17]. Under the assumption that the original image f and the noise $\mathcal{N}_\alpha$ are independent and that $\mathcal{N}_\alpha$ follows a Gaussian distribution, the variance of the degraded image can be written as follows

$$\alpha^2_{g(x,y)} = \alpha^2_{f(x,y)} + \alpha^2_{\mathcal{N}_\alpha},$$

where $\alpha^2_{g(x,y)}$ and $\alpha^2_{f(x,y)}$ are local variances. According to this equation, everywhere the original image is locally constant and void of edges (*i.e.* $\alpha^2_{f(x,y)} = 0$), one can assume that

$$\alpha^2_{g(x,y)} = \alpha^2_{\mathcal{N}_\alpha}.$$

In this way, whenever a region in $g(x,y)$ is uniform, the variance over that region should be close to that of the noise. Of course, the accuracy of the estimated noise variance is deeply bound to ones' ability of localizing uniform regions. Since such approach is sensitive to outliers in textured areas, these methods often overestimate the variance of noise.

An alternative way of estimating the variance of noise $\alpha^2_{\mathcal{N}_\alpha}$ is by taking the minimum, the average or the median of the local variance estimated at every pixel [17]–[19]

$$\begin{aligned}
(\alpha^2_{\mathcal{N}_\alpha})_{min} &= \min_{(x,y)}(\alpha^2_{g(x,y)}),\\
(\alpha^2_{\mathcal{N}_\alpha})_{mean} &= \operatorname*{mean}_{(x,y)}(\alpha^2_{g(x,y)}),\\
(\alpha^2_{\mathcal{N}_\alpha})_{med} &= \operatorname*{median}_{(x,y)}(\alpha^2_{g(x,y)}).
\end{aligned}$$

However, as reported by Martin-Fernandez *et. al* [18], the min operator underestimates the noise variance, the mean operator overestimates it, and the median operator gives some intermediate results. They also mention that all three methods are somewhat ineffective when the noise level is low. The authors thus proposed an intermediate solution which involves a free parameter $0 \leq \lambda \leq 1$:

$$\alpha^2_{\mathcal{N}_\alpha} = \lambda(\alpha^2_{\mathcal{N}_\alpha})_{mean} + (1-\lambda)(\alpha^2_{\mathcal{N}_\alpha})_{min}, \qquad (1)$$

where λ gives a relative influence to the min and the mean operators. The authors compared results for $\lambda = 0.25, 0.5$, and 0.75.

More complex approaches involving singular value decomposition [20] and fuzzy logic [21] have been proposed. A popular method is the one by Donoho and Johnstone [22] which uses high-frequency wavelet coefficients:

$$\alpha^2_{\mathcal{N}_\alpha} = \alpha^2_{MAD} = \frac{\text{MAD}(y^{HH}_{xy})}{0.6745}, \qquad (2)$$

where y^{HH}_{xy} are the coefficients of the finest diagonal subband and MAD stands for the median absolute deviation. Another wavelet-based method has been proposed by Starck and Murtagh [23]. Their approach uses an *à trous* wavelet transform to locate pixels that do not contain any significant

signal, *i.e.* pixels with a background value plus some noise. The value of a pixel (x, y) is considered to be pure noise if its corresponding wavelet coefficients are significantly low. Since they assume that the background is mostly contained in the coarsest scale of the wavelet transform, the coarsest scale is subtracted from the original image. From that new image, the $\alpha^2_{\mathcal{N}_\alpha}$ is computed by considering the wavelet coefficients whose value is larger than $k\alpha_j$ (they choose $k = 3$) where α_j is the standard deviation of the noise at each scale j. The α_j values are estimated by taking the wavelet transform of a Gaussian-noise image with $\alpha = 1$. The authors show that the method works well on astronomical images.

b) Blur estimation: Blur identification is another important step for image restoration techniques. Here, the degradation process is represented by a convolution $g = f * h_\beta$, where g is a blurred version of f, and h_β is the to-be-estimated point-spread function (PSF).

A simple way for estimating blur given f and g is through the use of an homomorphic filter [9], [24]. Given that a convolution in the spatial domain corresponds to a point-wise multiplication in the spectral domain *i.e.*

$$\Im\{g\} = \Im\{f * h_\beta\} = FH.$$

h_β can be computed as follows :

$$h_\beta \quad = \quad \Im^{-1}\left\{\exp\left(\ln[G] - \ln[F]\right)\right\}$$

since $\ln[G] = \ln[F] + \ln[H]$. Note that $\Im$ stands for the Fourier transform operator and G, F and H are the spectral versions of f, g and h_β. Let us mention that although the PSF can be obtained with a simple division in the spectral domain (a so-called *inverse* filtering):

$$h_\beta \quad = \quad \Im^{-1}\left\{\frac{G}{F}\right\}.$$

Such method rarely produces any good results [9]. The reason being that whenever F has zeros or very small values at some frequency (u, v), the ratio $\frac{G(u,v)}{F(u,v)}$ dominates over the rest of the spectrum. As suggested by Chitale and Padgett [8] a work around to inverse filtering is the Wiener deconvolution filter. Although the Wiener filter has been designed to recover f given g and h_β, one can deconvolute g with f to obtain the PSF h_β.

Numerous blind blur estimation techniques have also been proposed. Most of these techniques estimate the PSF based on the location of zeros in the Fourier spectrum of g [25]–[29]. The PSF can be computed globally or by combining many local PSFs estimated on small portions of the image [30]. In this case, the global PSF is obtained by averaging the power spectrum of each local PSF. The Fourier-based approaches are computationally efficient and require minimal assumptions on the input image. However, it is well known that Fourier techniques require g to have a large signal-to-noise ratio unless zeros are difficult to localize. As a solution, Chang *et al.* [31] detect zero-crossing in the "bispectrum" of the g instead of its power spectrum. The authors argue that bispectrum suppresses additive Gaussian noise and thus, facilitates the detection of zeros in low signal-to-noise ratio images.

A related blind approach based on singular vectors and singular values was proposed by Devcic and Loncaric [32]. The singular value decomposition (SVD) of the degraded image is first computed in the spatial domain. Then, the DFT is applied on the obtained SVD matrix. The PSF is estimated from the frequential singular vectors, while noise variance is estimated from the smallest frequential singular values. Finally, the spectrum of original image singular vectors is estimated using exponential model of covariance function for which the spectrum of singular vectors of the degraded image is computed.

In [33], Elder and Zucker proposed an edge-based method to estimate blur. They proposed an image compression scheme based on edge and blur information only. Their method estimates two quantities: the intensity at edge locations in the image and the blur at those locations. This estimation is performed under the assumption that blurred edges can be characterized by a sigmoidal intensity gradient.

Besides deconvolution, other applications such as shape from defocus depend on blur estimation. In this case, due to the limited depth of field of a lens, blur is proportional to the distance between the camera and the scene [34]–[36]. In this case, estimating blur amounts to estimating depth. Usually, the defocusing process is modeled by the convolution of a perfectly focused image with a PSF whose size is proportional to depth. Assuming a Gaussian PSF, Pentland introduced a Fourier-based algorithm working on local patches [37]. Similar techniques working in spatial and spectral domain have been proposed [34], [38]–[42].

IV. The MS-SSIM factors

As mentioned previously, the MS-SSIM index [43] is based on three multiscale factors: 1) the luminance distortion (ld) 2) the contrast distortion (cd) and 3) the structure distortion (sd) between an image f and a degraded version of it g. The philosophy behind MS-SSIM lies in its definition of an image. For MS-SSIM, an $N \times M$ image is a point in the $\mathbb{R}^{MxN}$ space where any distortion is modeled by a translational vector added to a reference image. In that space, the length of the translational vector is proportional to the magnitude of the distortion. The two vectors responsible for luminance and contrast distortion span a plane on which lies the reference image. The authors mention that distortions corresponding to a rotation of that plane are associated to structural changes between f and g.

From its basic formulation, the luminance distortion at scale i is defined as

$$LD_i(f, g) = \frac{2\mu_f\mu_g + C_1}{\mu_f^2 + \mu_g^2 + C_1},$$

where μ_f and μ_g represent the mean intensity of f and g at scale i, and C_1 is a constant to avoid instability when $\mu_f^2 + \mu_g^2 \approx 0$. According to Weber's law [44], the magnitude of a just-noticeable luminance change δL is proportional to the background luminance L. In that case, $\mu_f = \gamma\mu_g$, where γ represents the ratio of the luminance of g versus f. Thus, the

luminance distortion can also be defined as

$$LD_i(f,g) = \frac{2\gamma\mu_f^2 + C_1}{(1+\gamma^2)\mu_f^2 + C_1}. \quad (3)$$

Contrast distortion at scale i is defined in a similar way:

$$CD_i(f,g) = \frac{2\alpha_f\alpha_g + C_2}{\alpha_f^2 + \alpha_g^2 + C_2}, \quad (4)$$

where C_2 is a non negative constant and α_f (resp. α_g) represents the standard deviation of f (and g) at scale i.

As for structure distortion at scale i, it is measured after subtracting the average luminance and normalizing the contrast of both f and g. This leads to:

$$SD_i(f,g) = \frac{2\alpha_{f,g} + C_3}{\alpha_f^2\alpha_g^2 + C_3}, \quad (5)$$

where $\alpha_{f,g} = \frac{1}{N-1}\sum_{i=1}^{N}(f_i - \mu_f)(g_i - \mu_g)$, and C_3 is a small constant.

Finally, the three MS-SSIM features are computed as follows :

$$ld(f,g) = [LD_M(f,g)]^{\alpha_M} \quad (6)$$

$$cd(f,g) = \prod_{i=1}^{M}[CD_i(f,g)]^{\beta_i} \quad (7)$$

$$sd(f,g) = \prod_{i=1}^{M}[SD_i(f,g)]^{\gamma_i}, \quad (8)$$

where the luminance comparison $LD_M(f,g)$ is computed only at the largest scale M. The three exponents α_M, β_i and γ_i are used to adjust the relative importance of different components. In this paper, $M = 5$ corresponds to the maximum scale, while $i = 1$ corresponds to the original resolution of the image. In [43], the authors have defined $\alpha_M = 1$ and $\beta_1 = \gamma_1 = 0.0448$, $\beta_2 = \gamma_2 = 0.2856$, $\beta_3 = \gamma_3 = 0.3001$, $\beta_4 = \gamma_4 = 0.2363$, and $\beta_5 = \gamma_5 = 0.1333$. Also, $C_1 = 0.01 \cdot L^2$ and $C_2 = C_3 = 0.03 \cdot L^2$ where L is the maximum image luminance, here 255.

According to Eq. (6), (7), and (8), a pair (f,g) (where g is a degraded version of f) is associated to a 3D point $(ld, cd, sd) \in [0,1]^3$.

V. Noise and Blur Distortions

Reducing the noise and enhancing sharpness in images is often critical to producing clear, high dynamic range images. However, good denoising and deconvolution techniques need to know the distortion process as well as the noise and blur parameters. A degradation process frequently accounted for is the one associated to most digital cameras [9]. According to this process, the lens of the camera induces blur and the digitizer adds random noise. It is well known that if the degradation process is linear and position invariant, and that the 3D scene is made of objects located roughly at the same distance from the camera, then the degraded image $g(x,y)$ is obtained as follows [9]:

$$g(x,y) = h_\beta * f(x,y) + \mathcal{N}_\alpha, \quad (9)$$

where h_β is a low-pass filter, $\mathcal{N}_\alpha$ is noise and $*$ indicates convolution. It is generally assumed that $\mathcal{N}_\alpha$ is an uncorrelated white noise associated to thermally generated electrons that build up in the CCD (other types of noise caused by physical interferences are neglected here). In this paper, we estimate different types of blur and noise all driven by one parameter. For the blur, we consider a zero-mean Gaussian filter (defined by a standard deviation) and a uniform low-pass filter (defined by its size). As for noise, we consider zero-mean Gaussian noise (defined by a standard deviation) and salt and pepper noise (defined by the percentage of corrupted pixels). In this way, whatever the combination of noise and blur that we ought to estimate, we end up estimating two parameters: α and β.

Now that the distortion model between f and g has been introduced, the connexion between the (α,β) space and the MS-SSIM space becomes straightforward. By combining Eq. (6), (7), (8), and (9), one can associate a pair (α,β) to a 3D MS-SSIM point as follows:

$$(\alpha,\beta) \rightarrow (l,c,s), \quad (10)$$

where

$$l = ld(f, h_\beta * f + \mathcal{N}_\alpha),$$
$$c = cd(f, h_\beta * f + \mathcal{N}_\alpha),$$
$$s = sd(f, h_\beta * f + \mathcal{N}_\alpha).$$

In order to illustrate this relation, we took 676 points arranged on a 26×26 lattice in the (α,β) space and mapped it to the MS-SSIM space following Eq. (11). These points go from $(0,0)$ to $(\alpha_{\text{MAX}}, \beta_{\text{MAX}})$ and specify the amount of blur and noise. As shown in figure 1, whatever the content of f, the 3D points form a surprisingly smooth manifold. This strongly suggests that a change in the (α,β) distortion space induces a predictive change in the MS-SSIM space. More specifically, we empirically observed that

1) there is a one-to-one mapping between each (α,β) pair and its 3D correspondence (ld, cd, sd);
2) the manifolds have two principal directions corresponding to variations of α and β.

Of course, given the highly non-linear aspect of the MS-SSIM features, it is difficult (and maybe impossible) to formally prove the one-to-one mapping assertion for all non-trivial images[2]

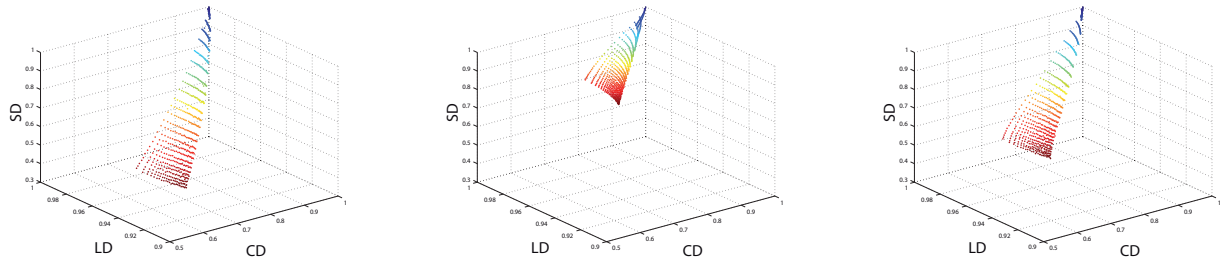
That being said, we conducted a study on 58 images sampled from the LIVE [45] and the COREL [46] image databases in order to establish an empirical demonstration that would support our hypothesis. Our demonstration involves the smallest distance between two neighbors in the MS-SSIM space as a function of the number of samples in the parameter space. We distorted each image using Eq. (9) given $n \times n$ (α,β) points. Then, following Eq. (11), we projected these $n \times n$ image pairs to the MS-SSIM space. This lead to a 3D scatter plot for each image similar to the ones shown in Fig. 1. For each projected point, we computed the distance to the nearest neighbor. Fig. 2 shows this distance averaged over all images and all sampling resolution as n increases.

[2]The one-to-one mapping assertion does not hold for perfectly uniform images since $h_\beta * f(x,y) = f(x,y) \quad \forall\beta$. However, that situation is very specific as the distortion process does not modify the content of such image.

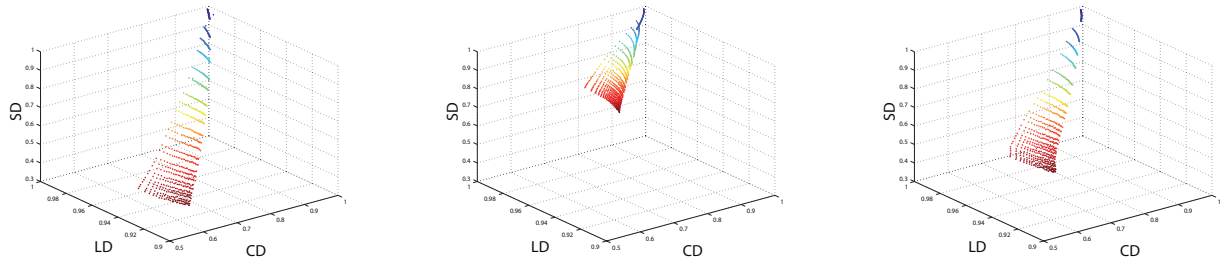
(a) Test image *bikes*.

(b) Test image: *house*.

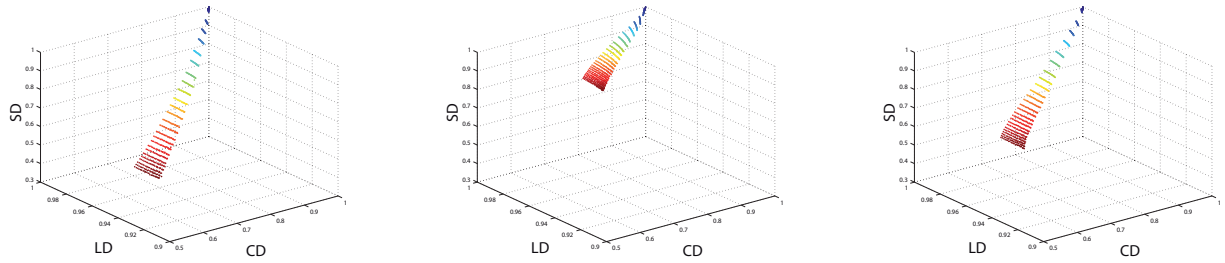
(c) Test image: *cemetery*.



(d) Manifolds for Gaussian blur and Gaussian noise.



(e) Manifolds for average blur and Gaussian noise.



(f) Manifolds for Gaussian blur and salt & pepper noise.

Fig. 1. **MS-SSIM manifolds obtained from three different images. Each point in the (ld,cd,sd) space corresponds to a specific (α, β) distortion. These manifold correspond to a combination of (d) Gaussian blur and Gaussian noise, (e) average blur and Gaussian noise, and (f) Gaussian blur and salt & pepper noise.**

As can be seen from the shape of the curve, the minimum distance between two neighboring points in the MS-SSIM space is never zero even for a 100×100 grid. In fact, the curve shows an asymptotic shape and never reaches zero. This clearly shows that, even though collisions may theoretically occur, the probability that two parameter pairs $X_1 = (\alpha_1, \beta_1)$ and $X_2 = (\alpha_2, \beta_2)$ project at the same location in the MS-SSIM space decreases asymptotically as the distance between X_1 and X_2 increases. This result allows us to conclude that the one-to-one mapping holds for arbitrary points X_1 and X_2 in the parameter space.

VI. Distortion Estimation as an Optimization Problem

As mentioned previously, our goal is to estimate the amount of noise α^* and blur β^* contained in g given f. Given the bijective transformation between the (α, β) space and the MS-SSIM space, the distortion estimation procedure can be formulated as an optimization procedure (the reader can follow the upcomming exposition through figure 3). Assuming that the pair (f, g) corresponds to a unique 3D point (lc^*, cd^*, sd^*) in the MS-SSIM space (the round dot figure 3), the goal is to find a pair $(\hat{\alpha}, \hat{\beta})$ such that $(f, f * h_{\hat{\beta}} + \mathcal{N}_{\hat{\alpha}})$ corresponds to a 3D point $(\hat{lc}, \hat{cd}, \hat{sd})$ (the square dot in figure 3) located as close as possible to (lc^*, cd^*, sd^*). Estimating (α^*, β^*) thus becomes

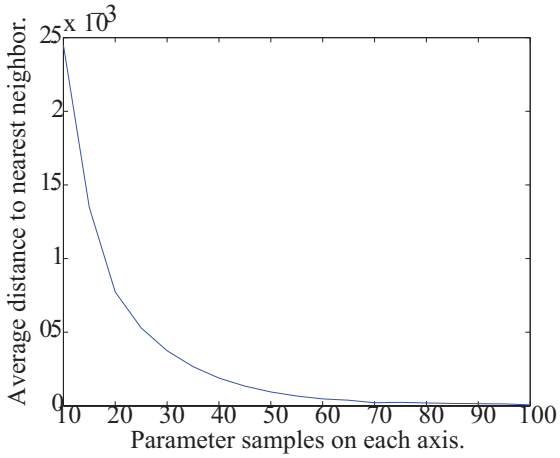


Fig. 2. **Relationship between the distance to nearest neighbors in the parameter space and the same distance in MS-SSIM space. The horizontal axis corresponds to the number of points on each axis in the parameter space. The vertical axis corresponds to the average distance for all images in our database to the nearest neighbor in the MS-SSIM space.**

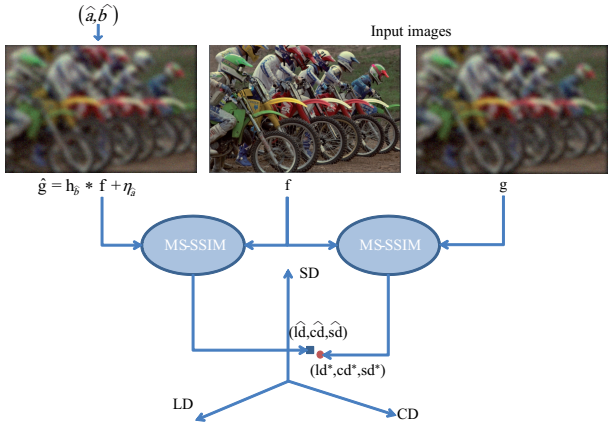


Fig. 3. **Schematic representation of our optimization procedure.**

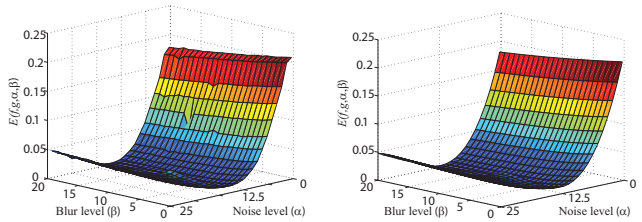
an optimization problem that we formalize as follows:

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} E(f, g, \alpha, \beta), \tag{11}$$

where $\alpha \in [0, \alpha_{\text{MAX}}], \beta \in [0, \beta_{\text{MAX}}]$, and $E(.)$ is an Euclidean distance $\sqrt{(lc - lc^*)^2 + (cd - cd^*)^2 + (sd - sd^*)^2}$. In order to illustrate the shape of this error function, we computed its value over 10000 samples (α, β) given that the solution (α^*, β^*) is $(\frac{\alpha_{\text{MAX}}}{2}, \frac{\beta_{\text{MAX}}}{2})$. As shown in figure 4(a), the error function is globally smooth and has a global minima in the middle. Unfortunately, since the 3D shape of $E(f, g, \alpha, \beta)$ is unknown *a priori*, gradient-descent optimizers are not applicable here. In fact, such problem is a so-called unconstrained optimization problem without derivatives [47]. Three search strategies adapted to that problem are introduced in the next subsections.

A. Brute Force Search (BF)

The simplest way to recover $(\hat{\alpha}, \hat{\beta})$ given (f, g) is by considering a large number of (α, β) values and keep the one whose 3D point (lc, cd, sd) is the closest to (lc^*, cd^*, sd^*) (*i.e.* the one with the lowest error $E(.)$). Of course, the more samples considered, the more precise the end result will be. As one would expect, considering a large number of (α, β) values (here 676) is prohibitive computational wise. The reason being that computing (lc, cd, sd) with $f * h_\beta + \mathcal{N}_\alpha$ followed by Eq.(6), (7), and (8) is a time consuming procedure.



(a) Error function $E(f, g, \alpha, \beta)$. (b) Error function $E(f, g, \alpha, \beta)$ with the use of a bicubic patch.

Fig. 4. **Error function E for the *Bikes* image without and with the use of a bicubic patch. The minimum is at $(\frac{\alpha_{\text{MAX}}}{2}, \frac{\beta_{\text{MAX}}}{2}) = (12.5, 10.0)$.**

B. Simplex Search (SI)

To reduce the computational effort, one needs to visit a smaller number of (α, β) values before reaching the global minimum. The simplex search is a typical solution to such unconstrained derivative-free optimization problems [48]. Simplex starts with 3 solutions called *vertices* which form a triangle in the (α, β) space. New positions are then iteratively identified inside and outside the triangle. The error value at these new positions is then compared to the vertices of the triangle. Then, the vertice with the highest cost is displaced such that its cost decreases. This leads to a new triangle and a new iteration. This operation is repeated until one of the vertices' cost is below a predefined threshold or when the number of points visited has reached some maximum value.

C. NewUOA (NU)

Although the Simplex algorithm is a classical solution to unconstrained optimization without derivatives, more recent developments aim at faster convergence, better numerical stability and improved robustness with respect to more complex objective functions.

Trust-region optimization methods are a family of derivative-free optimization techniques using an approximation of the objective function in order to reduce the cost of evaluations. This is important in applications where evaluating the objective function is costly. A typical trust region method [49] consists in building a good approximation of the objective function local to a trusted region, followed by a series of minimizations over this model. At each iteration, a new candidate point is generated by minimizing the approximation. Either the model is good and this minimization predicted a good candidate, in which case the new point is adopted and the region of trust grows, or the local approximation is judged inaccurate, in which case the size of the trusted region is reduced and the model is updated.

The NewUOA optimization software [50] is a recent development using quadratic approximations. The method presents

implementation tricks for reducing the number of interpolation conditions, further reducing the number of objective function evaluations, as well as enhancing numerical stability. Experimental results reported in [50] show that NewUOA has good performance for functions of up to 160 variables, yet drastically reducing the number of evaluations of the objective function when compared to other optimizers based on quadratic models.

VII. Fast 2D to 3D Mapping

So far, we introduced two optimizers (simplex and NewUOA) whose processing time is drastically faster than the simple brute force search (benchmarks are provided in Section IX). This is because simplex and NewUOA pick less points in the (α, β) space than the brute force search does. In this section, we introduce approximations of the manifold to further reduce the processing time.

We showed in figure 1 that the (α, β) space form a smooth manifold in the MS-SSIM space. This manifold has two principal directions associated to α and β. As shown in figure 5, the position of a 3D point on the manifold is tightly bound to the (α, β) values. Clearly, a 3D point on the MS-SSIM manifold is determined by the magnitude of α and β. Stated that way, α and β can be seen as parameters allowing to navigate on the 3D manifold. In other words, a good parametrization of the $(\alpha, \beta) \leftrightarrow (ld, cd, sd)$ mapping could allow to predict where a (α, β) point falls in the MS-SSIM space. Such parametrization would allow to map a distortion pair (α, β) to its related (ld, cd, sd) position without having to compute Eq. (6), (7), (8), and (9). This would reduce quite significantly the computational effort.

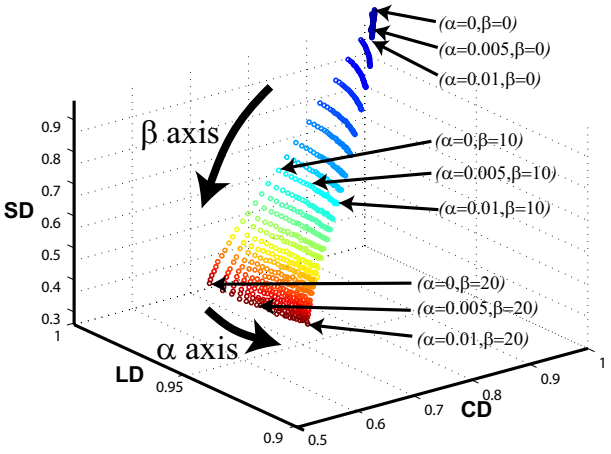


Fig. 5. **MS-SSIM manifold for the *cemetery* image (third image in figure 1). The manifold can be seen as a parametric patch on which every point is defined by a (α, β) value.**

Our mapping function is inspired by the shape of the 3D manifolds. As can be seen in figure 1 and 5, the 3D manifolds have a shape close to the one of a parametric patch [51]. In a similar fashion, the $(\alpha, \beta) \leftrightarrow (ld, cd, sd)$ mapping is also very close to the mathematical definition of a parametric patch. A patch is a mathematical function relating two parameters $(s, t) \in [0, 1]^2$ to a 3D point $(x, y, z) \in \mathbb{R}^3$:

$$(x(s,t)\ y(s,t)\ z(s,t))^{\mathrm{T}} = S.M.G.M^{\mathrm{T}}.T^{\mathrm{T}}. \qquad (12)$$

For a bi-cubic patch, $S = (1\ s\ s^2\ s^3)$, $T = (1\ t\ t^2\ t^3)$, G is a 4×4 matrix containing 16 control points, and M is the 4×4 *basis matrix* defining the nature of the patch (a Bezier, an Hermite, or any other patch). The manifolds in the (ld, cd, sd) space being smooth, one can fit a patch on it with a Vandermonde matrix[3] and with 16 control points obtained after uniformly sampling the (α, β) space with a 4×4 lattice (see Appendix 1 for more details on how matrix G is built). Once G has been filled, any pair (α, β) can be mapped to the MS-SSIM space as follows :

$$\begin{aligned} (s,t) &= (\alpha/\alpha_{\text{MAX}}, \beta/\beta_{\text{MAX}}) \\ (ld, cd, sd) &= S.M.G.M^{\mathrm{T}}.T^{\mathrm{T}}. \end{aligned} \qquad (13)$$

Such a patch fits surprisingly well on the manifold. As can be seen in figure 4 (b), the error function $E(f, g, \alpha, \beta)$ obtained with a bicubic patch has a very smooth shape close to the original error function.

A. Patch-Based Optimization Procedures

The use of a patch allows to map a 2D point (α, β) to its 3D MS-SSIM position (lc, cd, sd) with little computational effort (only 4 matrix multiplications). Since it does not change the optimization function of Eq. (11), the optimization procedures proposed so far can account for this mapping without having to change their functionality. Only the $(\alpha, \beta) \leftrightarrow (ld, cd, sd)$ mapping procedure need to be changed. We tested two such patch-based optimizers namely **PBF** (patch-based brute force search) and **PSI** (patch-based simplex search).

B. Newton-Raphson Search (NR)

The use of a parametric patch allows to formulate the problem in a different way: given a 3D point $A = (lc, cd, sd)$ associated to (f, g), find its projection (s, t) on the patch such that the distances between A and $B(s,t) = (x(s,t), y(s,t), z(s,t))$ is minimum. In other words, find the best (s, t) such that the Euclidean distance between A and $B(s,t)$ (namely $F = ||A - B(s,t)||$) is minimum. Unfortunately, when dealing with bicubic patches, there is no closed-form solution to that problem as it requires to find the roots of a fifth-degree polynomial. A solution [52] is to assign an initial approximation of s and t and solve it using a conventional fixed-point scheme. Since we want to minimize $||A - B(s,t)||$ we assume that the best (s, t) is the one for which $\frac{\partial}{\partial s}||A - B(s,t)|| = 0$ and $\frac{\partial}{\partial t}||A - B(s,t)|| = 0$. According to the Newton-Raphson formula:

$$s^{[k+1]} = s^{[k]} - \frac{F_s}{F'_s} \qquad t^{[k+1]} = t^{[k]} - \frac{F_t}{F'_t},$$

where k is an iterator, $F'_s = \frac{\partial}{\partial s}||A - B||$ and $F'_t = \frac{\partial}{\partial t}||A - B||$. multiplying s and t by α_{MAX} and β_{MAX}.

[3] $M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & 1/9 & 1/27 \\ 1 & 2/3 & 4/9 & 8/27 \\ 1 & 1 & 1 & 1 \end{pmatrix}^{-1}$.

C. Refined Patches

Algorithm 1 Refined Patches

Input: f,g,α_{MAX},β_{MAX}
Output: $\hat{\alpha}$,$\hat{\beta}$
1: $\alpha_1 = \beta_1 = 0$
2: $\alpha_2 = \alpha_{\text{MAX}}$, $\beta_2 = \beta_{\text{MAX}}$.
3: **for** $i = 0$ to iterMax **do**
4: $G \leftarrow$ 16 MS-SSIM points covering $[\alpha_1, \alpha_2] \times [\beta_1, \beta_2]$.
5: $(\hat{\alpha}, \hat{\beta}) \leftarrow$ Optimizer(f,g,G) /*PBF, PSI, or NR*/
6: $\hat{\alpha} = \hat{\alpha}(\alpha_2 - \alpha_1) + \alpha_1$
7: $\hat{\beta} = \hat{\beta}(\beta_2 - \beta_1) + \beta_1$
8: $\alpha_1 = \max(\hat{\alpha} - \frac{\alpha_{\text{MAX}}}{4^i}, 0)$
9: $\alpha_2 = \min(\hat{\alpha} + \frac{\alpha_{\text{MAX}}}{4^i}, \alpha_{\text{MAX}})$
10: $\beta_1 = \max(\hat{\beta} - \frac{\beta_{\text{MAX}}}{4^i}, 0)$
11: $\beta_2 = \min(\hat{\beta} + \frac{\beta_{\text{MAX}}}{4^i}, \beta_{\text{MAX}})$
12: **end for**

Although bicubic parametric patches fit well the MS-SSIM manifolds, they are nonetheless approximations and can be slightly inaccurate. One way of reducing these inaccuracies is through the use of so-called *refined patches*. The concept behind the refined patches is the same for every patch-based optimizer. Whenever a patch-based optimization method converges toward a solution $(\hat{\alpha}, \hat{\beta})$ (be it PBF, PSI, or NR), the goal is to fit a smaller (and therefore more accurate) patch in the vicinity of $(\hat{\alpha}, \hat{\beta})$ and re-start the optimizer on that localized patch. This procedure iterates a pre-specified number of times. In our experiments, numerical precision prevents any gain past 3 iterations. As shown in Algorithm 1, the search space defined by (α_1, β_1) and (α_2, β_2) reduces at each iteration.

Note that because of the nature of the patch, the $(\hat{\alpha}, \hat{\beta})$ values returned by the optimizer ranges between 0 and 1. These values thus need to be remapped into the (α, β) space. This is done at line 6 and 7.

VIII. Blind Distortion Estimation

So far, we presented a method for estimating the amount of distortion given known distortion types (say, Gaussian noise and Gaussian blur). But one question arises when the distortion types are unknown *a priori*. Fortunately, following Eq.(11), the answer to this question is fairly straightforward (the reader can follow the exposition through Algo.2). Let P be a set of noise and blur distortion types. For example, P can contain a combination of Gaussian, average and Butterworth noise, and Gaussian, average, salt-and-pepper noise. Given P, the goal is to find for each distortion pair $p \in P$ their associated parameters $(\hat{\alpha_p}, \hat{\beta_p})$ following one of the optimization procedure presented so far. Once every $(\hat{\alpha_p}, \hat{\beta_p})$ have been estimated, the one with the lowest global energy $E(f, g, \hat{\alpha_p}, \hat{\beta_p})$ is retained. As can be seen in Fig. 6, this algorithm makes the assumption that the error function $E(.)$ is lower for the true distortion type than for any other distortion.

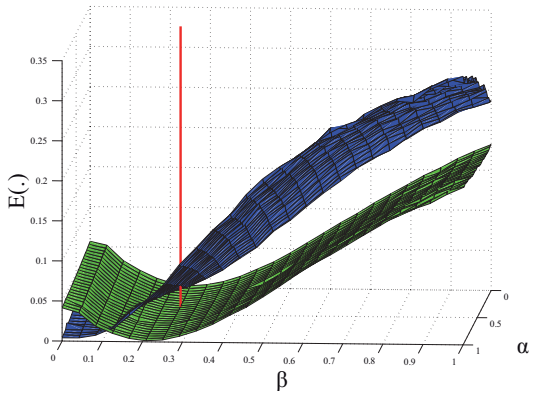


Fig. 6. **Error function E for image *bikes* given two distortion types: Gaussian blur & Gaussian noise and average blur and salt-and-pepper noise. The true distortion parameters (α, β) are pinpointed with the vertical line. The energy is significantly lower around the red line for the true distortion type (here Gaussian blur and Gaussian noise) than for the other one.**

Algorithm 2 Blind Distortion Estimation

Input: f,g,P
Output: α_{BEST}, β_{BEST},p_{BEST}
1: $\alpha_{\text{BEST}} = \beta_{\text{BEST}} = 0$; $E_{\text{MIN}} = \infty$
2: **for** each distortion $p \in P$ **do**
3: $(\hat{\alpha}_p, \hat{\beta}_p) \leftarrow$ find parameters given process p.
4: $\hat{E} \leftarrow E(f, g, \hat{\alpha_p}, \hat{\beta_p})$
5: **if** $\hat{E} < E_{\text{MIN}}$ **then**
6: $E_{\text{MIN}} = \hat{E}$
7: $\alpha_{\text{BEST}} = \hat{\alpha}$; $\beta_{\text{BEST}} = \hat{\beta}$
8: $p_{\text{BEST}} = p$
9: **end if**
10: **end for**

In order to validate this procedure, we computed a confusion matrix based on the returned value p_{BEST} (the chosen distortion process) compared to the actual distortion process used to compute g from f. Three noise and blur combinations were chosen namely:

A: gaussian filter, gaussian noise
B: average filter, salt and pepper noise
C: motion blur, multiplicative noise (speckle)

Tests were performed for a 15×15 grid in the parameter space for all of our 58 test images and BFS has been used as optimization procedure. Table I presents the resulting confusion matrix. The results present a clear dominance on the diagonal indicating that Algo.2 reliably chooses the appropriate distortion process.

	A	B	C
A	71.3%	14.2%	14.5%
B	9.8%	83.1%	7.1%
C	6.0%	6.2%	87.8%

TABLE I
CONFUSION MATRIX FOR DISTORTION PROCESS SELECTION USING ALGO 2.

IX. Results

A. Experimental apparatus

In order to gauge performances, we tested our seven optimization methods on 58 real-life images taken from the LIVE [45] and the COREL databases [46]. These images, whose size ranges between 610×488 and 768×512, are distorted with different amount of noise and blur. For every degraded image, the estimated amount of distortion $(\hat{\alpha}, \hat{\beta})$ is compared to the actual amount (α, β) and the error expressed as a percentage of the respective intervals $[0, \alpha_{\text{MAX}}]$ and $[0, \beta_{\text{MAX}}]$:

$$E(\hat{\alpha}) = \frac{|\hat{\alpha} - \alpha|}{\alpha_{\text{MAX}}}, \quad E(\hat{\beta}) = \frac{|\hat{\beta} - \beta|}{\beta_{\text{MAX}}}.$$

In our experiments, $\alpha_{\text{MAX}} = 25$ and $\beta_{\text{MAX}} = 20$. The first method we implemented is the brute force search technique. In this case, the (α, β) search space is sampled with a regular 26×26 lattice ranging from $(0, 0)$ to $(\alpha_{\text{MAX}}, \beta_{\text{MAX}})$. The second method is the simplex search for which we choose the Nelder-Mead algorithm [48]. The initial triangle is centered in the middle of the search space at position $(\frac{\alpha_{\text{MAX}}}{2}, \frac{\beta_{\text{MAX}}}{2})$ and the algorithm stops when 50 points have been visited. As for the NewUOA method, we used the code provided by the authors [53]. As recommended in the original paper [49], 5 points were used for the quadratic interpolation and the procedure is initialized at $(\frac{\alpha_{\text{MAX}}}{2}, \frac{\beta_{\text{MAX}}}{2})$. The confidence region has a radius of 1 and the algorithm stops when the distance to the objective is less than 0.1 or when 50 evaluations have been made.

As mentioned in section VII-C, the four other methods use a bi-cubic patch. The Newton-Raphson search reaches convergence when both $|s^{[k+1]} - s^{[k]}|$ and $|t^{[k+1]} - t^{[k]}|$ are below 0.0001. The initial approximation of s and t is obtained by averaging the s and t values of the 4 nearest control points. As for PBF and PSI, they use the same parameters as their non-patch version. We also tested the refined patch strategy on the PSI technique. We called that method RPSI.

We compared our seven methods to four state-of-the-art techniques. Two of these methods addresses the noise estimation problem (one blind and one non-blind) and two addresses the blur estimation problem (one blind and one non-blind). These four methods recover Gaussian noise and Gaussian blur.

The blind noise estimation method is a fast version [6] of Buckley's thin-plate smoothing spline method [7] and is referred to as the *EVAR* method. The method removes from g an estimated version of f ($\hat{f}$) obtained by fitting patches on g. The variance of noise is then estimated as follows : $\text{var}(\hat{f} - g)$. The non-blind noise estimation procedure directly removes f from g and computes variance : $\text{var}(f - g)$. We called this method *VAR*.

As for the blind blur estimation method, we used a Lucy-Richardson [54], [55] maximum likelihood PSF estimation algorithm which we call *LR-PSF* [56], [57]. This algorithm restores the image and estimates the PSF simultaneously following an iterative procedure. The maximum number of iterations is set to 20. Although this algorithm requires no knowledge on the shape of h, it nonetheless requires a size to be given. Since the size of h cannot be estimated based on g only, it is set to the largest expected PSF. Since $\beta_{\text{MAX}} = 20$ and that the size of a Gaussian filter must be at least $6 \times \beta$ [9], the PSF size is set to 120. Note that the `deconvblind()` function in the Matlab image processing toolbox have been used for LR-PSF. As for the non-blind blur estimation method, we retained the homomorphic filter described in section III. We call this method *HMF*. Details regarding the 11 methods are summarized in Table II.

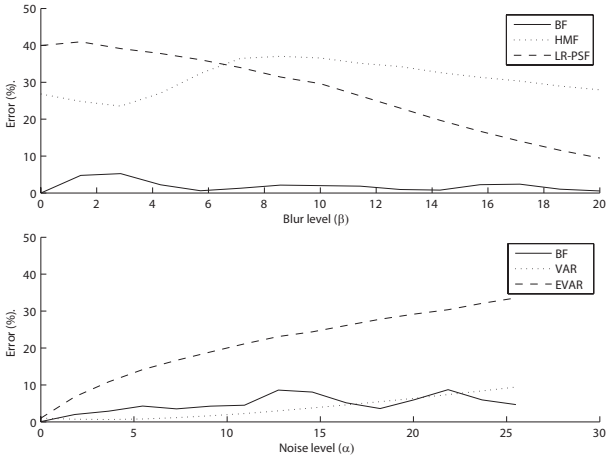


Fig. 7. **Comparison between our BF method and four state-of-the-art techniques for a single distortion. The plots show the average error across all 58 images for (top) blur estimation and (bottom) noise estimation.**

B. Results

c) Single Distortion: The first round of tests involves single distortions. We first degraded all 58 images with various amounts of Gaussian blur ranging from $\beta = 0$ to $\beta = \beta_{\text{MAX}}$. Then, for each β value, we computed the average error across all images for our BF method, LR-PSF and HMF. We then tested BF, VAR and EVAR for various amounts of Gaussian noise ($\alpha = 0.01$ to $\alpha = \alpha_{\text{MAX}}$). Again, the average error for each α value across all images has been computed for each method. Results are shown in figure 7.

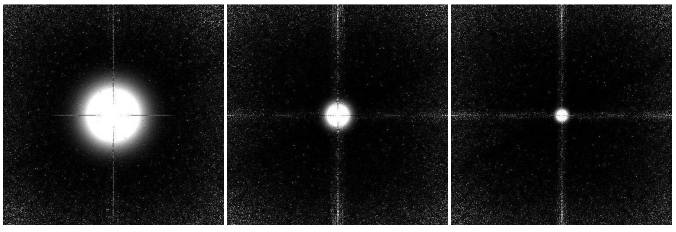


Fig. 8. **Three PSFs estimated by HMF. These PSF are in the spectral domain and correspond to $\alpha = 2, 5$ and 10. The white spikes distributed in high frequencies are caused by the log of near-zero amplitudes.**

As can be seen in the first plot, our method completely outperforms LR-PSF and HMF as it constantly produces errors below 5%. Also, LR-PSF performs better on large PSFs than on smaller ones. This is due to the filter size that we fixed to 120, a size better suited to large PSFs. As for HMF, although not blind, it does not perform well. The reason

Achronym	Name	Parameters
BF	Brute force search	(α, β) space sampled with a regular 26×26 lattice.
SI	Simplex search	Initial triangle centered at $(\frac{\alpha_{\text{MAX}}}{2}, \frac{\beta_{\text{MAX}}}{2})$; 50 points maximum visited.
NU	NewUOA search	5 points used for interpolation; confidence region's radius equals 0.25; algorithm starts at $(\frac{\alpha_{\text{MAX}}}{2}, \frac{\beta_{\text{MAX}}}{2})$; algorithm stop when distance to the objective < 0.0001 or when the number of evaluations reaches 50.
NR	Newton-Raphson search	Initial approximation of s and t : average of the s and t values of the 4 nearest control points; Algorithm stops when $\lvert s^{[k+1]} - s^{[k]} \rvert$ and $\lvert t^{[k+1]} - t^{[k]} \rvert < 0.0001$.
PBF	Patch-based brute force search	use of a bi-cubic patche + BF parameters.
PSI	Patch-based simplex search	use of a bi-cubic patche + SI parameters.
RPSI	Refined Patch-based simplex search	iterMax = 3; use of a bi-cubic patche + SI parameters.
EVAR	Blind noise-estimation method	None
VAR	Non-blind noise-estimation method	None
LR-PSF	Lucy-Richardson PSF estimation method	Filter size = 120×120, iterMax = 20.
HMF	Homomorphic filter	Filter size = 120×120

TABLE II
DISTORTION ESTIMATION ALGORITHMS TESTED IN THIS PAPER.

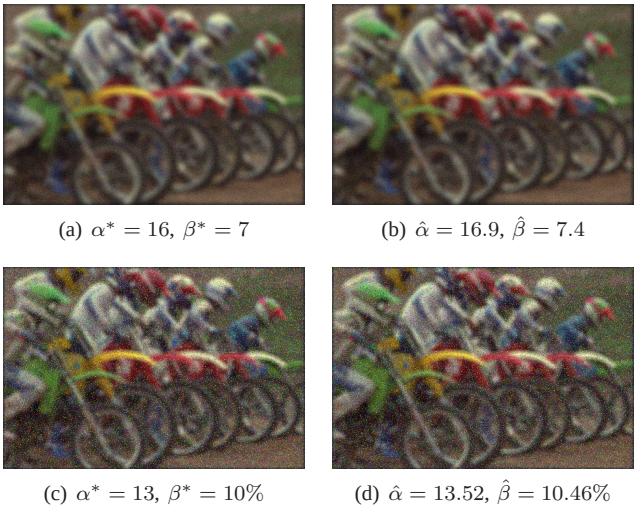
for this is related to the log operator which is sensitive to low-amplitude frequencies. This is illustrated in figure 8 in which three PSFs estimated by HMF are shown in the spectral domain. As can be seen, bright spikes caused by log of near-zero values are scattered in high frequencies. Although our implementation tries to filter out these values, they nonetheless bias the estimated PSFs.

The second plot shows noise estimation results. Without any surprise, our method does better than the blind estimation method EVAR as its performance decreases with noise. This can be explained by the fact that EVAR relies on its ability of correctly estimating f. However, the thin-plane spline strategy is not good at recovering f from g when g contains fine texture corrupted with a large amount of noise. As for VAR, our method does better for noise levels above 18. This is somehow contradictory since VAR is mathematically the best possible estimate and should always be very accurate. But as shown in the plot, the error increases almost linearly with α. This unexpected behavior can be explained by the fact that pixel values in a digital images are limited between 0 and 255. Because of that, the degradation process implicitly implements the following function:

$$g(x, y) = \min(255, \max(0, h_\beta * f(x, y) + \mathcal{N}_\alpha)), \qquad (14)$$

where $\min$ and $\max$ are clamping operators. In this way, the noise contained in the residual image $f - g$ follows a clipped Gaussian distribution whose standard deviation is different than that of $\mathcal{N}_\alpha$. As α increases, more and more values are clipped, inducing more errors in the estimation at higher levels. Our method does not suffer from this effect because it implicitely incorporate the clipping distortion operators.

d) Combined Distortions: Here, all 58 images are applied 225 Gaussian distortions (α, β) linearly distributed between $(0, 0)$ and $(\alpha_{\text{MAX}}{=}25, \beta_{\text{MAX}}{=}20)$ for a total of 13050 degraded images. The overall results are presented in figure 9 in which the average error (and standard deviation) for every method is presented.



(a) $\alpha^* = 16$, $\beta^* = 7$ (b) $\hat{\alpha} = 16.9$, $\hat{\beta} = 7.4$

(c) $\alpha^* = 13$, $\beta^* = 10\%$ (d) $\hat{\alpha} = 13.52$, $\hat{\beta} = 10.46\%$

Fig. 10. **Typical results obtained by a search strategy (here PBF) for (a),(b) Gaussian noise and Gaussian Blur and (c),(d) salt & pepper noise and average filter blur. Given an image f (here *Bikes*) and a degraded image g whose degradation parameters are $(16.0, 7.0)$ and $(13.0, 10\%)$, the recovered parameters ($(16.7, 7.4)$ and $(13.52, 10.17\%)$) give a new figure $\hat{g}$ that is very similar to g.**

As far as precision is concerned, BF produces the best results among our seven methods with 1.9% error for blur estimation and 7.52% error for noise estimation. Note that those results could be further improved with more samples. But although precise, BF is very slow as it is 18 times slower than the second slowest method. In fact, all other methods are significantly faster than BF. Of the four patch-based methods, NR produces slightly less precise results than the other ones. A surprising observation is that PSI is more precise than SI. This can be explained by the fact that patches make the to-be-minimized error function smoother and void of local minima (see figure 4). As we expected, the results further improved with a refined strategy (here RPSI) although at the cost of

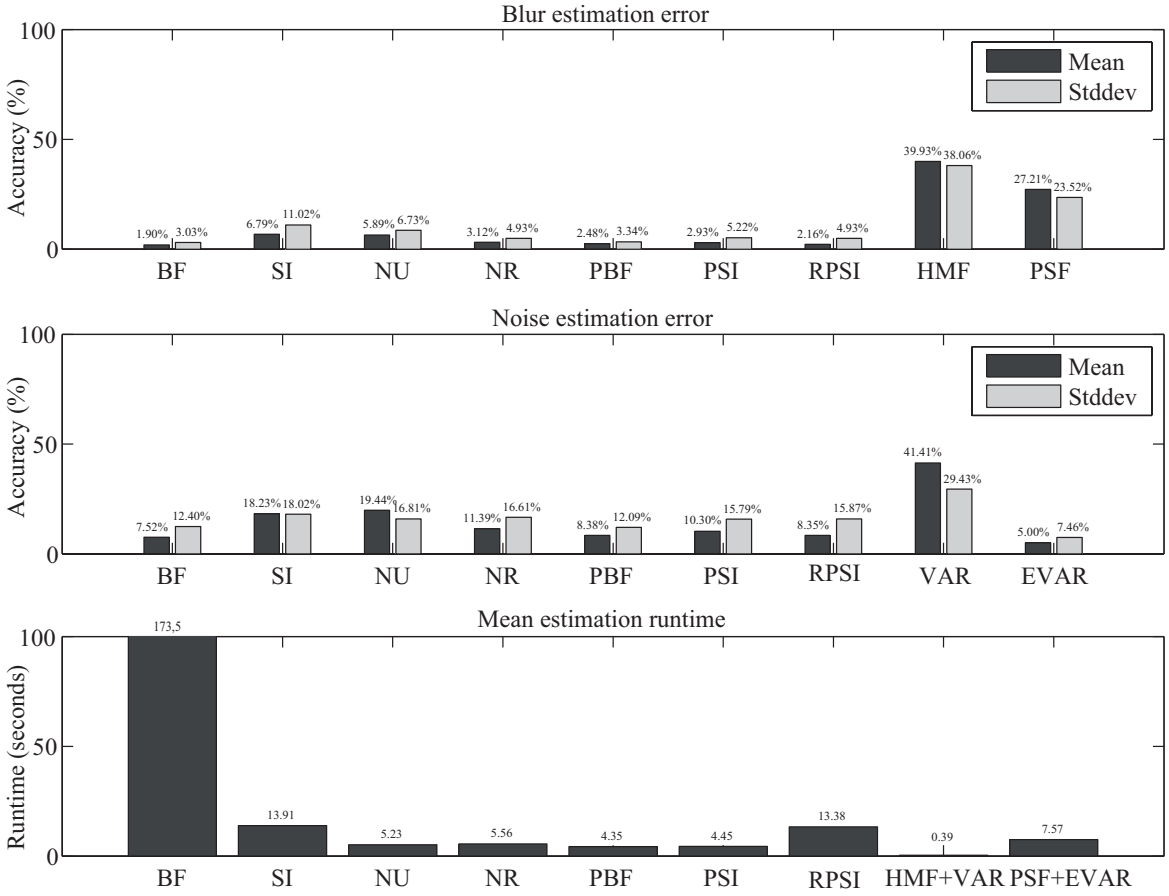


Fig. 9. **Global comparison between our seven methods and four state-of-the-art methods for the estimation of Gaussian blur and Gaussian Noise. The first and second plots show the average (and standard deviation) blur and noise estimation error. The third plot shows the average runtime for each method. Every method has been executed in Matlab.**

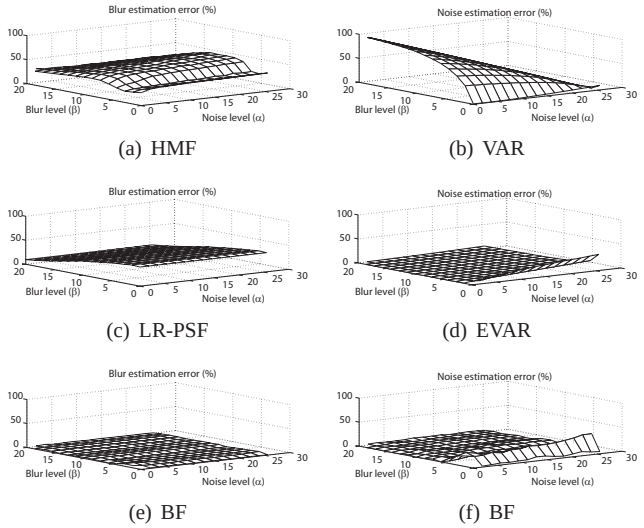
a larger computational effort. This clearly underscores the fact that patches not only reduces processing time, but also help reducing error rates. As for NewUOA and simplex, they both produced roughly 6% and 19% error for noise and blur estimation which is slighly worst than our other methods. Overall, when considering all seven methods, PBF offers in our view the best compromise between speed (58 times faster than BF), precision and conceptual simplicity.

As for the other four methods, we noticed with little surprise that HMF, PSF and VAR and not well suited to recover blur and noise when more than one degradation is applied. However, EVAR has a low error rate of 5.0% which was quite unexpected for a blind-estimation method. To better understand how EVAR (and the other methods) globally performs, we plotted the average error for each degradation value (α, β). This lead to the 3D surfaces shown in figure 11.

As can be seen in figure 11 (d), the performance of EVAR increases with blur. Although counterintuitive, this behavior is normal since EVAR aims at estimating α according to the following function : $g(x, y) = f(x, y) + \mathcal{N}_\alpha(x, y)$. However, when the input image is degraded with blur, $f(x, y)$ is in fact a smooth image $f(x, y) * h_\beta(x, y)$. As far as EVAR is concerned, estimating a smooth image $f(x, y) * h_\beta(x, y)$ with a spline-based method is far easier than estimating $f(x, y)$ alone. This explains the high accuracy of EVAR in figure 11.

When considering figure 11 (e) and (f), one can see that BF is well suited to blur but has a hard time estimating small amounts of noise. The reason being that the manifold in the MS-SSIM space is usually cramped on the noise axis near the origin $(1, 1, 1)$ as shown in figure 1. High proximity of values in these areas hinder the brute force algorithm's performance. To solve that problem, a distance other than Euclidean should be considered. However, we left to future work the development of such distance function.

We also tested our method on two other distortions, namely a salt-and-pepper noise and a box-filter blur. Here again, all 58 images were applied 225 distortions (α, β) linearly distributed between $(0, 0)$ and (α_{MAX}=21, β_{MAX}=20%) for a total of 13050 degraded images. In this case, β stands for the size of the box filter and α for the percentage of corrupted pixels. Results are presented in figure 12 in which the average error (and standard deviation) for each method is presented. As can be seen, results are slightly less precise for blur and roughly the same for noise estimation than those obtained for the estimation of Gaussian blur and Gaussian noise. This being said, the processing time stayed globally the same. The brute force search and its patch-based version are still the two most precise methods, followed by RPSI, NR, and PSI. Simplex and NewUOA are still the two least precise solutions with error rates above 10% and 15%. Since PBF is 6 times faster than RPSI, it is still in



(a) HMF (b) VAR

(c) LR-PSF (d) EVAR

(e) BF (f) BF

Fig. 11. **The average error of Gaussian blur and Gaussian noise estimation for all images computed over the entire (α, β) search space.**

our view the best compromise between speed and accuracy. Furthermore, such results clearly show that our framework is general enough to cope with different types of blur and noise distortions as can be seen in figure 10.

X. Conclusion

In this paper, we presented a method to simultaneously estimate the amount of blur and noise in a distorted image g given a reference image f. We have shown that, provided a multi-dimensional quality metric (MS-SSIM), distortion values (α, β) are associated to 3D points which form a manifold in the MS-SSIM space. We formalized the estimation problem as a search problem for which 7 different search algorithms have been proposed. The simple brute force search algorithm is the slowest approach but shows high precision. We also showed that replacing the mapping function by a patch drastically reduces the computation effort while improving results. It is the case for the simplex optimizer whose patch-based version is significantly more precise. This is due to the patch smoothing out the error function and eliminating local minima. Refined patch strategy prove to further improve precision while maintaining a substantial speedup with respect to brute force search. The NewUOA and Simplex techniques did not perform well globally although their processing time are way below BFS's.

In conclusion, we believe that the patch-based brute-force technique offers the best compromise between accuracy, speed and simplicity. Also, results obtained on different types of distortions shown that our method does well on non-Gaussian distortions. To our knowledge, this property is unique to our approach.

In the future, we look forward to test other types of distortion functions to include multiplicative noise, compression artifacts and non-linear blur functions.

Appendix 1

As mentioned in Section VII, the bicubic patch requires 16 control points stored in the 4×4 matrix G. Each of these control points is defined in the MS-SSIM space. In order to get these 3D control points, one first need to uniformly sample the (α, β) space as shown in Table III. Then, for each (α, β) pair, the image pair $(f, g = f * h_\beta + \mathcal{N}_\alpha)$ is computed. These image pair are then mapped to the MS-SSIM space following Eq.(6), (7), and (8) and stored in matrix G.

References

[1] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

[2] S. Rezazadeh and S. Coulombe, "A novel approach for computing and pooling structural similarity index in the discrete wavelet domain," in *Proc. IEEE Int. Conf. Image Processing*, 2009, pp. 2209–2212.

[3] D. M. Chandler and S. S. Hemami, "VSNR: A wavelet-based visual signal-to-noise ratio for natural images," *IEEE Trans. Image Process.*, vol. 16, no. 9, pp. 2284–2298, 2007.

[4] H. R. Sheik and A. C. Bovik, "A visual information fidelity measure for image quality assessment," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 430–444, 2006.

[5] S.-C. Tai and S.-M. Yang, "A fast method for image noise estimation using laplacian operator and adaptive edge detection," in *Int. Symp. on Comm. Cont. and Signal Proc.*, 2008, pp. 1077–1081.

[6] D. Garcia, "Robust smoothing of gridded data in one and higher dimensions with missing values," *Comput Statist Data Anal*, vol. 54, pp. 1167–1178, 2010.

[7] M. J. Buckley, "Fast computation of a discretized thin-plate smoothing spline for image data," *Biometrika*, vol. 8, no. 2, pp. 247–258, 1994.

[8] S. Chitale and W. T. Padgett, "Blur identification and correction for a given imaging system," in *Southeastcon*, 1999, pp. 268–273.

[9] R. C. Gonzalez and R. E. Woods, *Digital image processing, 2nd Ed.* Prince Hall, 2001.

[10] P. F. C. de Rivaz and N. G. Kingsbury, "Bayesian image deconvolution and denoising using complex wavelets," in *Proc. IEEE Int. Conf. Image Processing*, 2001, pp. 273–276.

[11] M. Figueiredo and R. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Trans. Image Process.*, vol. 12, no. 8, pp. 906–916, 2003.

[12] T. Treibitz and Y. Y. Schechner, "Recovery limits in pointwise degradation," in *IEEE Int. Conf. Computational Photography*, 2009.

[13] D.-H. Shin, R.-H. Park, S. Yang, and J.-H. Jung, "Block-based noise estimation using adaptive gaussian filtering," *IEEE Trans. Consum. Electron.*, vol. 51, no. 1, pp. 218–226, 2005.

[14] R. Kaspar and B. Zitova, "Weighted thin-plate spline image denoising," in *Int. Conf. Computer Analysis of Images and Patterns*, 2003, pp. 730–737.

[15] K. Rank, M. Lendl, and R. Unbehauen, "Estimation of image noise variance," in *Vis. Image Signal Process*, vol. 146, 1999, pp. 80–84.

[16] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising methods, with a new one," *SIAM Multiscale Modeling and Simulation*, vol. 4, no. 2, pp. 490–530, 2005.

[17] S. A. Fernandez, G. V. S. Ferrero, M. M. Fernandez, and C. A. Lopez, "Automatic noise estimation in images using local statistics. additive and multiplicative cases," *Image and Vision Computing*, vol. 27, pp. 756–770, 2009.

$t \backslash s$	0	$\frac{1}{3}$	$\frac{2}{3}$	1
0	$(0,0)$	$(\frac{\alpha_{\max}}{3},0)$	$(\frac{2\alpha_{\max}}{3},0)$	$(\alpha_{\max},0)$
$\frac{1}{3}$	$(0,\frac{\beta_{\max}}{3})$	$(\frac{\alpha_{\max}}{3},\frac{\beta_{\max}}{3})$	$(\frac{2\alpha_{\max}}{3},\frac{\beta_{\max}}{3})$	$(\alpha_{\max},\frac{\beta_{\max}}{3})$
$\frac{2}{3}$	$(0,\frac{2\beta_{\max}}{3})$	$(\frac{\alpha_{\max}}{3},\frac{2\beta_{\max}}{3})$	$(\frac{2\alpha_{\max}}{3},\frac{2\beta_{\max}}{3})$	$(\alpha_{\max},\frac{2\beta_{\max}}{3})$
1	$(0,\beta_{\max})$	$(\frac{\alpha_{\max}}{3},\beta_{\max})$	$(\frac{2\alpha_{\max}}{3},\beta_{\max})$	$(\alpha_{\max},\beta_{\max})$

TABLE III

16 (α,β) CONTROL POINTS USED TO COMPUTE MATRIX G. EACH (α,β) PAIR IS ASSOCIATED TO A (s,t) PARAMETRIC VALUE.

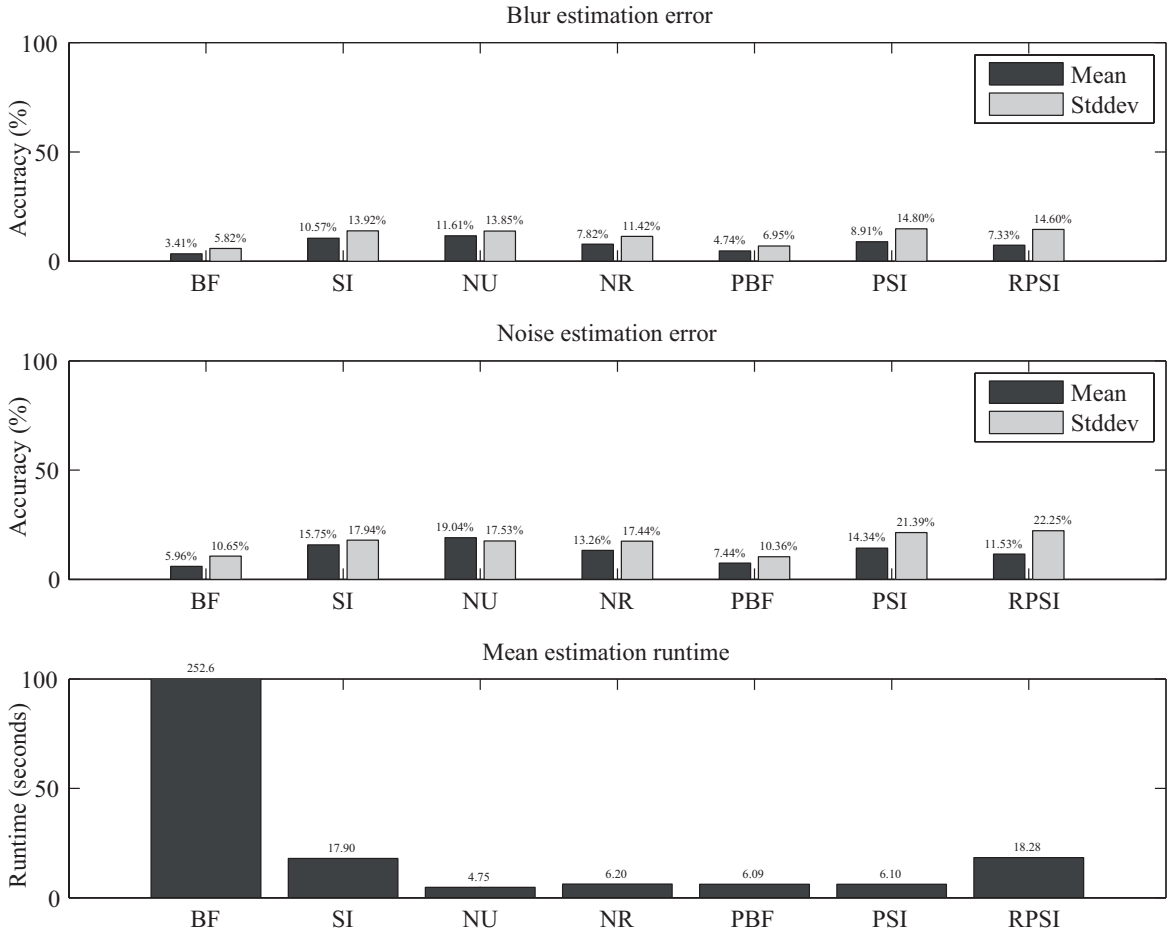


Fig. 12. **Global comparison between our seven methods and four state-of-the-art methods for low-pass box filter and salt-and-pepper noise. The first and second plots show the average (and standard deviation) blur and noise estimation error. The third plot shows the average runtime for each method. Every method has been executed in MATLAB.**

[18] M. Martn-Fernndez, R. San-Jos-Estpar, C. Westin, and C. Alberola-Lpez, "A novel gauss-markov random field approach for regularization of diffusion starting from tensor maps," *Lect. Notes Comput. Sci.*, vol. 2809, p. 506517, 2003.

[19] J. S. Lee and K. Hoppel, "Noise modeling and estimation of remotely sensed images," in *Int. Geoscience and Remote Sensing, Vancouver*, vol. 2, 1989, pp. 1005–1008.

[20] K. Konstantinides, B. Natarajan, and G. S. Yovanof, "Noise estimation and filtering using block-based singular value decomposition," *IEEE Trans. Image Process.*, vol. 6, no. 3, pp. 479–483, 1997.

[21] M. Salmeri, A. Mencattini, E. Ricci, and A. Salsano, "Noise estimation in digital images using fuzzy processing," in *Proc. IEEE Int. Conf. Image Processing*, 2001, pp. 517–520.

[22] D. Donoho and I. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, 1994.

[23] J. L. Starck and F. Murtagh, "Automatic noise estimation from the multiresolution support," *Publications of the Astronomical Society of the Pacific*, vol. 110, pp. 193–199, 1998.

[24] A. Oppenheim, R. Shafer, and T. J. Stockham, "Nonlinear filtering of multiplied and convolved signals," *Proc. IEEE*, vol. 16, no. 3, pp. 437–466, 1968.

[25] W. Che-Yen and L. Chien-Hsiung, "Point spread functions and their applications to forensic image restoration," *Journal of Forensic Science*, vol. 1, pp. 15–26, 2002.

[26] A. A. Bhutta and H. Foroosh, "Blind blur estimation using low rank approximation of cepstrum," in *ICIAR 2006, LNCS 4141*, 2006, pp. 94–103.

[27] M. Tico and M. Vehvilainen, "Estimation of motion blur point spread function from differently exposed image frames," in *EUSIPCO*, Florence, Italy, 2006.

[28] R. L. Lagendijk and J. Biemond, *Basic methods for image restoration and identification*, 2nd ed. Elsevier, Handbook of Image and Video

Processing, 2005, ch. 3.5, pp. 167–181.

[29] N. Joshi, R. Szeliski, and D. J. Kriegman, "PSF estimation using sharp edge prediction," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 2008.

[30] M. Cannon, "Blind deconvolution of spatially invariant image blurs with phase," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 24, pp. 58–63, 1976.

[31] M. Chang, A. Tekalp, and A. Erdem, "Blur identification using the bispectrum," *IEEE Signal Process. Lett.*, vol. 39, pp. 2323–2325, 1991.

[32] Z. Devcic and S. Loncaric, "Blind restoration of space-invariant image degradations in the singular value decomposition domain," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, 2001, pp. 49–52.

[33] J. H. Elder and S. W. Zucker, "Local scale control for edge detection and blur estimation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 7, pp. 699–716, 1998.

[34] Y. Xiong and S. Shafer, "Depth from focusing and defocusing," in *Proc. IEEE Conf. Computer Vision Pattern Recognition*, 1993, pp. 68–73.

[35] J. R. A. Torreao and J. L. Fernandes, "Single-image shape from defocus," in *Computer Graphics and Image Processing*, 2005, pp. 241–246.

[36] H. Jin and P. Favaro, "A variational approach to shape from defocus," in *Proc. European Conf. Computer Vision*, 2002, pp. 18–30.

[37] A. P. Pentland, "A new sense for depth of field," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 4, pp. 523–531, 1987.

[38] G. Schneider, B. Heit, J. Honig, , and J. Bremont, "Monocular depth perception by evaluation of the blur in defocused images," in *Proc. IEEE Int. Conf. Image Processing*, vol. 2, 1994, pp. 116–119.

[39] M. Subbarao and G. Surya, "Depth from defocus: A spatial domain approach," *Int. Journal of Computer Vision*, vol. 13, pp. 271–294, 1994.

[40] D. Ziou and F. Deschenes, "Depth from defocus in spatial domain," *Comp. Vision and Image Understanding*, vol. 81, pp. 143–165, 2001.

[41] A. Rajagopalan, S. Chaudhuri, and U. Mudenagudi, "Depth estimation and image restoration using defocused stereo pairs," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 11, pp. 1521–1525, 2004.

[42] P. Favaro, S. Soatto, M. Burger, and S. J. Oshe, "Shape from defocus via diffusion," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 30, no. 3, pp. 518–531, 2008.

[43] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multi-scale structural similarity for image quality assessment," in *IEEE Asilomar Conference on Signals, Systems, and Computers*, 2003, pp. 1398–1402.

[44] B. A. Wandell, *Foundations of vision*. Sunderland, Massachusetts: Sinauer Associates, 1995.

[45] L. C. H. R. Sheikh, Z. Wang and A. C. Bovik, "LIVE image quality assessment database release 2," *http://live.ece.utexas.edu/research/Quality*.

[46] The corel image database. [Online]. Available: http://wang.ist.psu.edu/docs/related/

[47] R. P. Brent, *Algorithms for minimization without derivatives*. Prentice-Hall: Englewood Cliffs, New Jersey, 1973.

[48] J. C. Lagarias, J. Reeds, M. Wright, and P. Wright, "Convergence properties of the nelder-mead simplex method in low dimensions," *SIAM J. on Optimization*, vol. 9, no. 1, pp. 112–147, 1998.

[49] K. Scheinberg, "Derivative free optimization method," IBM Watson Research Center, Tech. Rep., 2000.

[50] M. J. D. Powell, "The NewUOA software for unconstrained optimization without derivatives," in *Large-Scale Nonlinear Optimization*, vol. 83, 2006.

[51] L. Piegl and W. Tiller, *The NURBS book (2nd ed.)*. New York, USA: Springer-Verlag, Inc., 1997.

[52] M. Plass and M. Stone, "Curve-fitting with piecewise parametric cubics," *SIGGRAPH*, vol. 17, no. 3, pp. 229–239, 1983.

[53] NewUOA web page. [Online]. Available: http://www.inrialpes.fr/bipop/people/guilbert/newuoa/newuoa.html

[54] W. H. Richardson, "Bayesian-based iterative method of image restoration," *J. Opt. Soc. Am.*, vol. 62, no. 1, p. 5559, 1972.

[55] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *Astronom. J.*, vol. 79, no. 6, pp. 745–754, 1974.

[56] M. Williamson and A. Neureuther, "Utilizing maximum likelihood deblurring algorithm to recover high frequency components of scanning electron microscopy images," *J. Vac. Sci. Technol. B*, vol. 22, no. 2, pp. 523–527, 2004.

[57] D. Biggs and M. Andrews, "Acceleration of iterative image restoration algorithms," *Applied Optics*, vol. 36, no. 8, pp. 1766–1775, 1997.