Two Bayesian learning approaches to image processing

Yiqing Wang

Ecole Normale Supérieure de Cachan
Centre de Mathématiques et de Leurs Applications

Dissertation

submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy in Applied Mathematics

by

Yi-Qing Wang

Two Bayesian learning approaches to image processing

Thèse soutenue le 2 Mars 2015 devant le jury composé de :

M.	Boaz Nadler	Weizmann Institute of Science	Rapporteur
M.	Gabriel Peyré	CEREMADE, Université Paris-Dauphine	Rapporteur
M.	Jean-Michel Morel	Ecole Normale Supérieure de Cachan	Directeur
M.	Alain Trouvé	Ecole Normale Supérieure de Cachan	Président du jury
M.	Frédéric Guichard	DxO Labs	Examinateur
M.	Charles Kervrann	INRIA Rennes	Examinateur

Dedicated to my parents

Acknowledgements

I would like to express my profound appreciation and gratitude to my advisor Prof. Jean-Michel Morel for having given me the opportunity to embark on this enriching journey and providing me guidance and mentorship throughout the process. I have benefited enormously from his deep understanding of wide-ranging subjects including, but not limited to image processing. His passion and intuition for the discipline is what I have been trying to measure up to. Our weekly discussions and frequent email exchanges on my ongoing work as well as general research philosophy and methodologies will remain one of my fondest memories.

I am also indebted to Prof. Nicolas Vayatis for offering me to work as his teaching assistant for two years in the courses on the theoretical foundations of machine learning. It greatly expanded my knowledge of what the probability community has accomplished over the past few decades in understanding various data analysis tools. This role also added a unique dimension to my experience of academic life.

I would also like to thank Dr. Christophe Labourdette and Prof. Alain Trouvé for being so forgiving of my running a horrible amount of training programs on their CPU and GPU servers. In a similar vein, I would like to thank the exceptionally knowledgeable and talented Drs. Enric Meinhardt-Llopis and Nicolas Limare for being extremely nice and patient with my dumb questions.

CMLA's staff members Mmes. Véronique Almadovar, Micheline Brunetti, Sandra Doucet, Virginie Pauchont and Carine Saint-Prix certainly deserve special thanks. My days at the laboratory would not have been so smooth if it were not for their always timely help.

I am very grateful to Prof. Boaz Nadler and Dr. Gabriel Peyré for their time in reviewing this thesis and to Dr. Frédéric Guichard, Dr. Charles Kervrann and Prof. Alain Trouvé for agreeing to be my thesis committee members.

I appreciate the generosity of DxO Labs which provided part of the funding for my research.

Finally, to my parents, it is your love that gives me the strength to complete the present work.

Title Two Bayesian learning approaches to image processing

Abstract This work looks at two patch based image processing methods in a Bayesian risk minimization framework. We describe a Gaussian mixture of factor analyzers for local prior modelling and apply it in the context of image denoising and inpainting. We also study multilayer neural networks from a probabilistic perspective as a tool for conditional expectation approximation, which suggests ways to reduce their sizes and training cost.

Keywords Bayesian modelling, conditional expectation, mixture of factor analyzers, artificial neural network, image processing

Titre Traitement d'images par deux approches d'apprentissage Bayésien

Résumé Cette thèse porte sur deux méthodes à patch en traitement d'images dans le cadre de minimisation du risque Bayésien. Nous décrivons un mélange d'analyses factorielles pour modéliser la loi *à priori* des patchs dans une seule image et l'appliquons au débruitage et à l'inpainting. Nous étudions aussi les réseaux de neurones à multi-couches d'un point de vue probabiliste comme un outil permettant d'approcher l'espérance conditionnelle, ce qui ouvre quelques voies pour réduire leurs tailles et coût d'apprentissage.

Mots-clés modélisation Bayésienne, espérance conditionnelle, mélange d'analyses factorielles, réseau de neurones artificiels, traitement d'images

Contents

Introduction

0.1 Overview

Traditional image processing challenges such as denoising and inpainting constitute a vital testing ground for our understanding of the nature of digital images. Good solutions to these problems typically involve identifying and modelling natural image regularities, whether they lie in the spatial or transform domains. Historically, spatial domain methods such as level line Masnou and Morel (1998), total variation Rudin et al. (1992), Shen and Chan (2002), anisotropic diffusion Perona and Malik (1990), Bertalmio et al. (2000) and bilateral filter Tomasi and Manduchi (1998) all grew out of a desire to preserve image continuity, edges in particular.

The concept of regularity was then extended from pixels to image parts, or patches. Its advantage is twofold. First, typically square in form, patches form a standardized class of their own. With no restriction on their positions, patches can be viewed as random realizations from the same underlying distribution, thereby lending themselves naturally to statistical analysis. Second, depending on the size, patches can contain a flexible amount of information, rich enough for image representation and at the same time sufficiently limited for tractable numerical modelling. Nonlocal means Buades et al. (2005), for instance, is a bilateral filter acting on ℓ_2-adjacent patches.

In a parallel development, sparse representation has been in vogue ever since the appearance of wavelet analysis Daubechies et al. (1992), Mallat (2008). This line of research builds on a fundamental belief that some appropriately designed linear transforms can represent natural signals, including images, with just a few coefficients compared to their original size. Other transforms of this genre include steerable pyramid Simoncelli and Freeman (1995) and curvelet Candès (2003). Thanks to variational formulation, coefficient shrinkage Donoho and Johnstone (1994; 1995), and model based thresholding, they led to effective algorithms Portilla et al. (2003), Starck et al. (2002), Elad et al. (2005).

Advances in the field have since blurred the line separating spatial and transform methods. BM3D Dabov et al. (2007), for instance, uses spatial block matching to obtain similar patches, which are then jointly filtered in a fixed basis with coefficient shrinkage. In addition, how best to construct a basis has also been extensively investigated. Efficient optimisation techniques such as orthogonal matching pursuit Mallat and Zhang (1993) and the LARS algorithm Efron et al. (2004) allow to approach basis pursuit Chen et al. (1998) or Lasso Tibshirani (1996), which were formulated in an explicit quest for sparse coding and have since expanded into an umbrella

subject known as dictionary learning. This endeavor ultimately leads the research community to look beyond the image to restore itself and seek data driven priors. Various modelling tools including Gaussian mixture Zoran and Weiss (2011), 1 as well as unsupervised and supervised K-SVD Elad and Aharon (2006), Mairal et al. (2009; 2012) have been successfully employed.

Increasingly powerful and accessible computing facilities now make another line of research possible. Rather than try to summarize the analytically intractable patch space with dictionaries, one can process images using essentially an empirical replica of the patch space generated with a great number of samples from external image databases, which produced impressive results Levin and Nadler (2011) Yue et al. (2014). Moreover, with training cost no longer as prohibitive as it once was, researchers have also started to harness the potential of neural networks Bengio (2009). Convolutional neural networks Jain and Seung (2009), stacked sparse auto-encoder Xie et al. (2012) and plain multilayer neural networks Burger et al. (2012) all had success at image denoising.

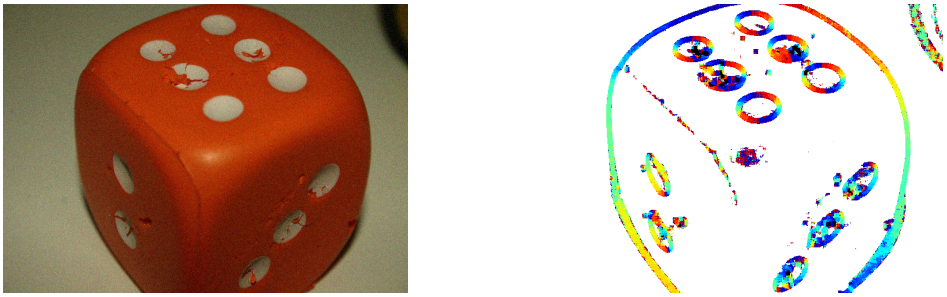
The rest of the thesis consists of four parts: the first two chapters describe an orientation based approach to patch prior modelling. The next three chapters look at multilayer neural networks as a device for conditional expectation approximation and derive applications from their interplay with the underlying patch distribution. The third section is centered on reproducible research and details the Viola-Jones algorithm, a successful application of the empirical risk minimization. Finally, the thesis ends with a review of parametric denoising algorithms from a Bayesian risk minimization perspective and some suggestions for future research.

0.2 Contributions of this thesis

The first part was inspired by evidence pointing to edge-like patterns as the most salient characteristics of relatively small image patches Olshausen and Field (1996), Takeda et al. (2007), Lee et al. (2006), Chatterjee and Milanfar (2009), Yu et al. (2012). Therefore, in order to reach the holy sparse land, it is essential to build a dictionary with basis vectors sporting similar features.

Gaussian mixture emerged as our modelling tool of choice as a universal density approximation device. As a result, patch classification is also straightforward thanks to the plug-in version of the Bayes rule Devroye et al. (1996). More importantly, equipped with the Expectation Maximisation algorithm, mixture can evolve and better account for image patches as data likelihood increases monotonously. Moreover, for both denoising and inpainting, numerical results show that a dynamic mixture can achieve similar or better performance as a static Gaussian mixture, despite being ten times smaller.

Chapter 1 presents the SURE (Stein's Unbiased Risk Estimator) guided Gaussian mixture image denoising algorithm capable of delivering state-of-the-art performance (see Figure 2) The algorithm fits a Gaussian mixture of factor analyzers to patches extracted from a single noisy image which, thanks to its orientation related interpretation, enables a visually accurate patch classification (see Figure 1).



(a) (b)

Figure 1 – *(a) noisy image ($\sigma = 10$) (b) patch orientation classification after one EM iteration. Patches assigned to different mixture components are marked in different colors. Those highlighted in white are detected as flat.*

When it comes to patch denoising, the algorithm switches between Wiener filter and hard shrinkage for individual mixture models according to a real time Bayesian risk tracking statistics (1) constructed with SURE (see Definition 0.1 and Theorem 0.1).

Definition 0.1 *Let $\tilde{P}$ be the sum of a κ-by-κ patch and a Gaussian random vector composed of i.i.d. entries distributed as $\mathcal{N}(0, \sigma^2)$. Let f be one of the following filters:*

1. *linear: $f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \langle \tilde{P} - \mu, b_j \rangle b_j + \mu$*
2. *soft shrinkage: $f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \gamma_t^{\textbf{soft}}\big(\langle \tilde{P} - \mu, b_j \rangle\big) b_j + \mu$ with $\gamma_t^{\textbf{soft}}(\omega) = sgn(\omega)(|\omega| - t)_+$*
3. *hard shrinkage: $f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \gamma_t^{\textbf{hard}}\big(\langle \tilde{P} - \mu, b_j \rangle\big) b_j + \mu$ with $\gamma_t^{\textbf{hard}}(\omega) = \omega 1_{|\omega| > t}$*

where $\mu, (c_j)_{1 \le j \le \kappa^2}, (b_j)_{1 \le j \le \kappa^2}$, and t are the filter's $\tilde{P}$-independent mean, coefficients, basis, and threshold. And their weak derivatives are defined to be

1. *linear: $\nabla \cdot f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j$*
2. *soft shrinkage: $\nabla \cdot f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j 1_{[t,+\infty)}\big(\big|\langle \tilde{P} - \mu, b_j \rangle\big|\big)$*
3. *hard shrinkage: $\nabla \cdot f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \big(1_{[t,+\infty)}\big(\big|\langle \tilde{P} - \mu, b_j \rangle\big|\big) + t\mathbb{E}\big[(\delta_t - \delta_{-t})\big(\langle \tilde{P} - \mu, b_j \rangle\big) | P\big]\big)$*

where $\delta_x(\cdot)$ represents a Dirac centered at $x \in \mathbb{R}$.

Theorem 0.1 *Under the assumptions of Definition 0.1, SURE given the observation $\tilde{P}$*

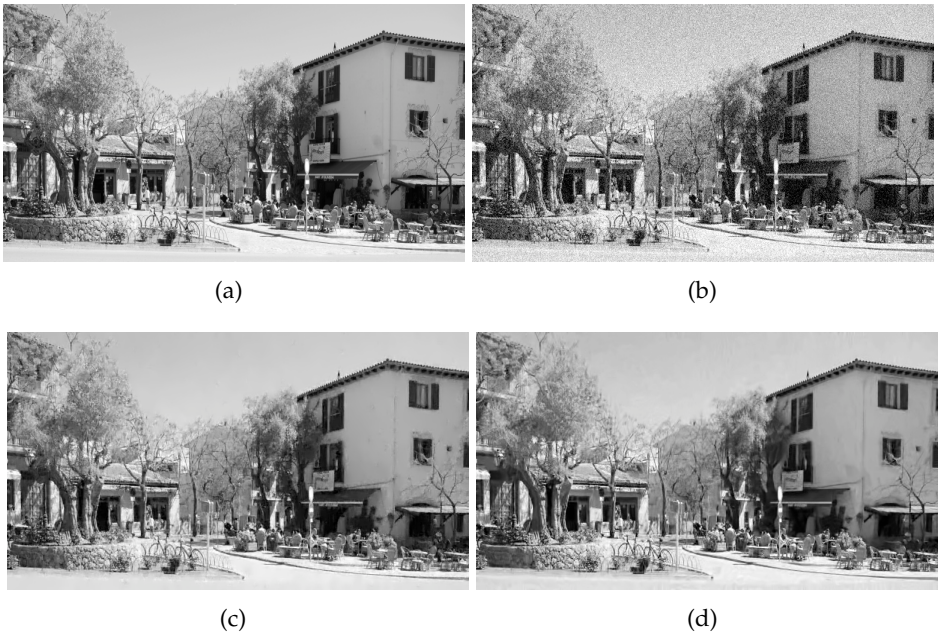
$$\underset{f}{\text{SURE}}(\tilde{P}) := \frac{1}{\kappa^2}\|\tilde{P} - f(\tilde{P})\|^2 - \sigma^2 + \frac{2\sigma^2}{\kappa^2}\nabla \cdot f(\tilde{P}) \tag{1}$$

is unbiased

$$\mathbb{E}\big[\underset{f}{\text{SURE}}(\tilde{P}) | P\big] = \mathbb{E}\big[\frac{1}{\kappa^2}\|P - f(\tilde{P})\|^2 | P\big].$$

Hence given a great number of patches $(\tilde{P}_i)_{1 \le i \le n}$, the Bayesian quadratic risk estimator $n^{-1}\sum_i \text{SURE}_f(\tilde{P}_i)$ may be used as a filter selection criterion. Moreover, thanks to Jensen's inequality, patch MSE can be established as

an asymptotic upper bound of image MSE, which makes our SURE based patch MSE estimate an algorithm stopping rule.



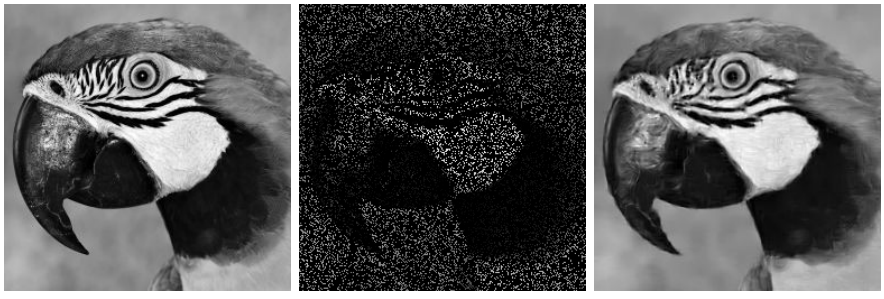
(a) (b)

(c) (d)

Figure 2 – *(a) original (b) noisy* σ=20 *(c) ours RMSE*=10.73 *(d) BM3D RMSE* =10.77

Chapter 2 is based on a namesake paper published in French, which is translated into English with more algorithm implementation details as Chapter 6. They document the existing piecewise linear estimate (PLE) algorithm for inpainting Yu et al. (2012) and propose several theoretical and numerical improvements among which, the most significant are

1. it uses a Gaussian factor mixture to put orientation models in a single unified probabilistic framework, which also accommodates for the first time an extra model for flat area detection;
2. it derives the associated Expectation Maximization (EM) procedure for parameter learning from partially observable data;
3. it applies tensor structure Harris and Stephens (1988) to mixture initialisation, thereby significantly accelerating the algorithm thanks to a reduced number of iterations required at parameter learning (see Figure 3).



(a) (b) (c)

Figure 3 – *(a) original (b) masked image (ratio*=0.8*) (c) one iteration E-PLE RMSE*=14.9

The second part focuses on multilayer neural networks and their applications to image processing. The fundamental observation is that driven by backpropagation, feedforward neural networks seek to approximate a conditional expectation. Though larger neural networks tend to be better, our work explores the possibility to reduce their sizes and training cost.

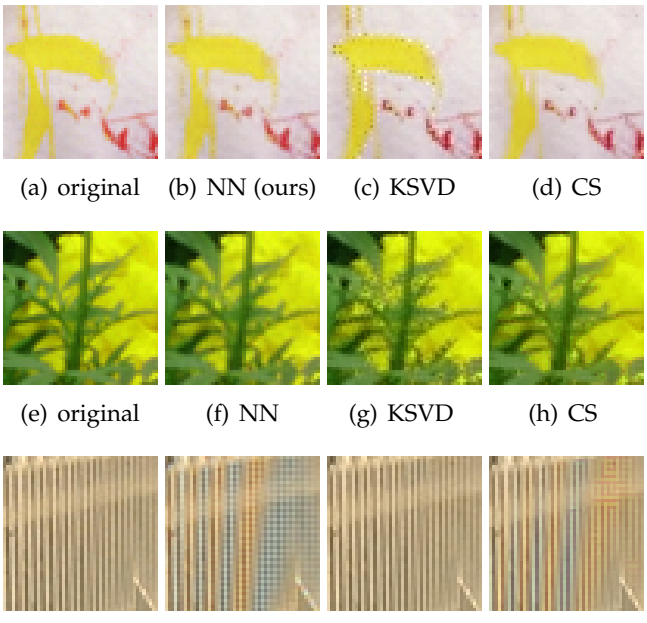
We have chosen image demosaicking and denoising because unlike random masks arising in the context of image inpainting, Bayer color filter array and additive Gaussian noise are two types of distortion with locally defined data format and thus fit well into the patch regression framework.

Chapter 3 presents a small two hidden layer neural network intended to estimate the masked 32 color intensity values from a 4-by-4 mosaiced patch. The neural network is designed to have an additive linear component so as to let its $\tanh(\cdot)$ enabled non-linear structure focus on approximating the difference. Because of the $\frac{\pi}{2}$-rotation-invariance of the Bayer color filter array (CFA) (see Figure 4) and image patch statistics,

(a) (b) (c) (d)

Figure 4 – *Four basic blocks of the Bayer pattern*

the neural network trained on the RGGB CFA pattern (see Figure 4(a)) can demosaick all four basic Bayer CFAs. It is shown (see Figure 5) that this neural network is competitive compared to the state-of-the-art algorithms which typically take longer to process a mosaiced image because of their larger input sizes.



(a) original (b) NN (ours) (c) KSVD (d) CS

(e) original (f) NN (g) KSVD (h) CS

(i) original (j) NN (k) KSVD (l) CS

Figure 5 – *Our neural network does not do well when image patterns to recover oscillate much. In contrast, it does interpolate well where color changes abruptly. Visually speaking, contour stencil Getreuer (2011a) yields the closest numerical results. However, the neural network seems to be more accurate on blobs.*

Given the impressive image denoising results obtained with multilayer neural networks Burger et al. (2012), Chapter 4 presents an attempt to reduce their training cost as these dedicated neural networks can only handle one level of Gaussian noise fixed at their respective training time and that their training may take months Burger (2013). Through an investigation of the patch distribution invariance with respect to linear transforms, we show how to make a single existing deep neural network work well across all levels of Gaussian noise (see Table 1).

Table 1 – *RMSE comparison between dedicated neural networks and our generic network*

$\sigma = 25$	**dedicated**	**generic**
computer	**7.99**	8.10
dice	2.77	2.77
flower	**4.59**	4.63
girl	**3.38**	3.41
traffic	**9.16**	9.22
valldemossa	**11.85**	11.99
avg.	**6.62**	6.68

$\sigma = 35$	**dedicated**	**generic**
computer	**9.63**	9.82
dice	**3.29**	3.45
flower	**5.53**	5.64
girl	**3.88**	4.07
traffic	**10.81**	10.94
valldemossa	**14.21**	14.28
avg.	**7.89**	8.03

$\sigma = 50$	**dedicated**	**generic**
computer	**11.59**	11.92
dice	**4.17**	4.50
flower	**6.85**	7.06
girl	**4.60**	4.92
traffic	**12.83**	13.07
valldemossa	**16.98**	17.06
avg.	**9.50**	9.75

$\sigma = 65$	**dedicated**	**generic**
computer	**13.16**	13.82
dice	**4.83**	5.39
flower	**7.89**	8.20
girl	**5.28**	5.74
traffic	**14.32**	14.78
valldemossa	**18.67**	18.99
avg.	**10.69**	11.15

$\sigma = 75$	**dedicated**	**generic**
computer	**14.10**	14.78
dice	**5.48**	6.14
flower	**8.57**	8.96
girl	**5.66**	6.20
traffic	**15.08**	15.56
valldemossa	**19.97**	20.34
avg.	**11.47**	11.99

$\sigma = 170$	**dedicated**	**generic**
computer	**20.51**	21.81
dice	**9.23**	10.92
flower	**12.84**	13.64
girl	**8.79**	10.54
traffic	**20.71**	21.72
valldemossa	**26.42**	27.26
avg.	**16.41**	17.64

From a probabilistic study of how different patch denoising algorithms approach condition expectation approximation, Chapter 5 presents a conditional expectation decomposition (see Theorem 0.2). It shows that estimation via conditional expectation mechanically involves likelihood ratio induced pattern classification and suggests a filter substitution strategy, whereby large neural networks' effective but computationally costly built-

in filters for structure and texture patterns are replaced by BM3D and small neural networks respectively.

Theorem 0.2 *Let $\mathbb{P}, \mathbb{M}, \mathbb{Q}$ be three probabilities defined on a common measurable space satisfying $\mathbb{P} = \alpha\mathbb{M} + (1-\alpha)\mathbb{Q}$ for some $\alpha \in (0,1)$. Let U, V be two random vectors with $\mathbb{E}_{\mathbb{P}}\|U\|_1 < +\infty$ defined on the same space. Then we have*

$$\mathbb{E}_{\mathbb{P}}[U|V] = \alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{M}}[U|V] + (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{Q}}[U|V], \quad \mathbb{P}\ a.e.$$

where $\frac{d\mathbb{M}}{d\mathbb{P}}$ and $\frac{d\mathbb{Q}}{d\mathbb{P}}$ are the Radon-Nikodym derivatives of $\mathbb{M}$ and $\mathbb{Q}$ with respect to $\mathbb{P}$.

The analysis and empirical studies result in a light-weight algorithm SSaNN, for self-similarity and neural network, that complements BM3D with small neural networks geared towards granular texture pattern denoising, thereby improving BM3D at minor additional cost (see Table 2).

Table 2 – *RMSE comparison between BM3D, small and large neural networks and our algorithm*

$\sigma = 10$	**small**	**BM3D**	**SSaNN**	**large**
computer	4.97	4.66	**4.63**	4.77
dice	2.58	**1.79**	**1.79**	1.89
flower	3.19	2.83	2.81	**2.77**
girl	2.93	**2.32**	**2.32**	2.41
traffic	5.67	5.64	5.54	**5.52**
valldemossa	6.55	6.61	6.51	**6.48**

$\sigma = 25$	**small**	**BM3D**	**SSaNN**	**large**
computer	8.88	8.15	**8.13**	8.20
dice	4.81	3.04	3.04	**2.85**
flower	5.61	5.15	5.12	**4.85**
girl	5.08	3.64	3.64	**3.46**
traffic	10.16	9.94	9.79	**9.55**
valldemossa	12.22	12.28	12.06	**11.85**

$\sigma = 35$	**small**	**BM3D**	**SSaNN**	**large**
computer	10.86	9.88	**9.87**	9.88
dice	6.24	3.80	3.80	**3.39**
flower	6.99	6.40	6.38	**6.00**
girl	6.39	4.41	4.41	**4.01**
traffic	12.26	11.85	11.75	**11.36**
valldemossa	14.81	14.74	14.53	**14.24**

Part I. Orientation based sparse patch modelling

The first part of this thesis is devoted to an orientation based approach to image processing. It was inspired by a host of evidence pointing to edge-like patterns as the most salient characteristics of relatively small image patches. Therefore, in order to reach the holy sparse land, it is essential to build a dictionary with basis vectors sporting similar features.

Gaussian mixture emerged as our modelling tool of choice because it is a universal density approximation device. As a result, patch classification is also straightforward thanks to the plug-in version of the Bayes rule. More importantly, equipped with the Expectation Maximisation algorithm, mixture can evolve and better account for image patches as data likelihood increases monotonously. For both denoising and inpainting, numerical results show that a dynamic mixture can achieve similar or better performance as a static Gaussian mixture, despite being ten times smaller.

1 SURE guided Gaussian mixture image denoising

In this chapter, using Gaussian factor modelling, its dedicated Expectation Maximization (EM) inference as well as a statistical filter selection and algorithm stopping rule, we develop a SURE (Stein's Unbiased Risk Estimator) guided image denoising algorithm capable of delivering state-of-the-art performance.

1.1 Gaussian factor for patch orientation modelling

When applied to approximate an arbitrary probability distribution, a Gaussian mixture's main appeal lies in its versatility and interpretability. Perhaps more importantly, it is equipped with the renowned Expectation Maximization (EM) algorithm Dempster et al. (1977), Wainwright and Jordan (2008) which makes its parameter inference particularly convenient.

Image patch distribution can be modelled with a Gaussian mixture. Unlike Zoran and Weiss (2011), we decided to assign to all the mixture components an orientation related interpretation because of the evidence presented in Olshausen and Field (1996), Takeda et al. (2007), Lee et al. (2006), Chatterjee and Milanfar (2009), Yu et al. (2012).

The covariance matrix of patches inclined at some angle θ, modelled as a Gaussian random vector, cannot be of full rank. If it were, all of its eigenvectors viewed as individual patches, would be oriented in either θ or $\pi - \theta$. As a result, none of their linear combinations could produce a patch with oriented at $\frac{\pi}{2} - \theta$, which contradicts the assumed full-rankness.

However, for practical purposes, a model should account for a slightly wider range of orientations than just θ: i.e. $(\theta - \Delta, \theta + \Delta)$ with a small scalar $\Delta > 0$. Thus, the above reasoning need not apply. But there is still reason to believe that a reduced set of vectors suffices for a good representation of patches of a narrow range of orientations. Therefore, instead of a full-fledged Gaussian distribution, an equally flexible model candidate in this case is a Gaussian factor model

$$P_\theta = \boldsymbol{F}_\theta c + \boldsymbol{\mu}_\theta.$$

The variability of the κ-by-κ patch P_θ can be adjusted through the l-column factor loading matrix $\boldsymbol{F}_\theta$. With $\boldsymbol{\mu}_\theta$ deterministic and c following $\mathcal{N}(0, I_l)$,

P_θ remains Gaussian. Note that the factor trimming also lowers the model complexity, which should help resist over-fitting.

In addition to these θ-indexed oriented models, two *non-oriented* models representing respectively textural and flat patches should be set up as well. If the resultant Gaussian mixture has K components, an i-indexed κ-by-κ noisy patch $\tilde{P}_i$ can be represented by

$$\tilde{P}_i = \sum_{k=0}^{K-1} (\boldsymbol{F}_k c_{ki} + \boldsymbol{\mu}_k + \sigma n_i) 1_{s_i=k}$$

where

1. $\boldsymbol{F}_k \in \mathbb{R}^{\kappa^2 \times l_k}$: a deterministic l_k-column matrix for the k-th model;
2. $c_{ki} \in \mathbb{R}^{l_k}$: a Gaussian coefficient distributed as $\mathcal{N}(0, I_{l_k})$;
3. $\boldsymbol{\mu}_k \in \mathbb{R}^{\kappa^2}$: a deterministic vector representing the k-th model mean;
4. $\sigma \in \mathbb{R}_+$: the standard deviation of zero-mean Gaussian noise;
5. $n_i \in \mathbb{R}^{\kappa^2}$: a noise vector following $\mathcal{N}(0, I_{\kappa^2})$ and independent of c_{ki};
6. $s_i \in \{0, \cdots, K-1\}$: a random model for the i-th patch.

For EM to succeed at its task, we try to set a good starting position in the parameter space by initializing the mixture using random patches drawn directly from natural images with the help of the *tensor structure* orientation detector Harris and Stephens (1988): given a patch P, the discrete gradient $\nabla P(r,u)$ is calculated at all pixel sites (r,u) in its domain $\mathrm{dom}(P)$. The patch's orientation v_* can then be defined by

$$\begin{aligned} v_* &= \underset{\|v\|_2=1}{\operatorname{argmin}} \sum_{(r,u)\in\mathrm{dom}(P)} \|\nabla P(r,u) - v^T \nabla P(r,u) v\|_2^2 \\ &= \underset{\|v\|_2=1}{\operatorname{argmin}} \sum_{(r,u)\in\mathrm{dom}(P)} \|\nabla P(r,u)\|_2^2 - (v^T \nabla P(r,u))^2 \\ &= \underset{\|v\|_2=1}{\operatorname{argmax}}\, v^T \Big(\sum_{(r,u)\in\mathrm{dom}(P)} \nabla P(r,u) (\nabla P(r,u))^T \Big) v. \end{aligned} \tag{1.1}$$

The solution to 1.1 is the first leading eigenvector of the positive semidefinite matrix (denoted by M_P) enclosed in the parentheses. Let λ_b and λ_s ($\lambda_b \geq \lambda_s$) be its eigenvalues. Because of the equality

$$\sum_{(r,u)\in\mathrm{dom}(P)} \|\nabla P(r,u)\|_2^2 = \mathrm{tr}(M_P) = \lambda_s + \lambda_b,$$

it seems natural to declare P oriented if

$$\frac{\sum_{(r,u)\in\mathrm{dom}(P)} \|\nabla P(r,u) - v_*^T \nabla P(r,u) v_*\|_2^2}{\sum_{(r,u)\in\mathrm{dom}(P)} \|\nabla P(r,u)\|_2^2} = \frac{\lambda_s}{\lambda_s + \lambda_b}$$

is small. We tuned a threshold $t_{\textbf{orient}} = 5$ empirically such that a patch is labelled oriented if and only if its eigenvalues satisfy

$$\lambda_s^{-1} \lambda_b \geq t_{\textbf{orient}} \tag{1.2}$$

Its orientation θ_* is then set to $\psi(\arctan \frac{y_*}{x_*})$ with $v_* = (x_*, y_*)^T$ and $\psi(a) = a 1_{a\geq 0} + (\pi + a) 1_{a<0}$ which ensures that θ_* is positive. To distinguish between two non-oriented categories, we use

$$\lambda_b \geq t_{\textbf{flat}} \quad \text{and} \quad \lambda_s^{-1} \lambda_b < t_{\textbf{orient}} \tag{1.3}$$

to define textural patches. The remaining set of patches satisfying

$$\lambda_b < t_{\textbf{flat}} \quad \text{and} \quad \lambda_s^{-1}\lambda_b < t_{\textbf{orient}} \tag{1.4}$$

are flat. Again, the threshold $t_{\textbf{flat}} = 10^4$ was selected empirically.

The definitions 1.2, 1.3, 1.4 split the first quadrant $(\lambda_s, \lambda_b) \in \mathbb{R}_+^2$ into three regions, with the one characterized by 1.2 further divided into $K-2$ sub-areas by angle quantification to form a K-zone partition (see Algorithm 1). More precisely, a patch P is assigned to the k-th oriented model if and only if it satisfies

$$\lambda_s^{-1}\lambda_b \geq t_{\textbf{orient}} \quad \text{and} \quad \theta_* \in [\frac{k}{K-2}\pi, \frac{k+1}{K-2}\pi).$$

Since λ_s, λ_b and θ_* all depend on P, the natural patch space is now endowed with a coarse σ-algebra entirely induced by the tensor structure.

Algorithm 1 Gaussian Mixture Initialization

Input: some clean natural grayscale images.
Parameter: number of mixture components K, patch dimension $\kappa \times \kappa$.
for $k = 0$ to $K - 1$ **do**
  $N_k \leftarrow 0$ with N_k the number of sample patches collected for the k-th model.
end for
while $\min_{0 \le k \le K-1} N_k < 5000$ **do**
  randomly draw a $\kappa \times \kappa$ patch P from the image set.
  solve 1.1 to deduce the eigenvalues (λ_b, λ_s) and the eigenvector v associated with λ_b.
  if $\lambda_b / \lambda_s < t_{\textbf{orient}}$ **then**
    if $\lambda_b < t_{\textbf{flat}}$ **then**
      assign P to the flat model.
      $N_{K-1} \leftarrow N_{K-1} + 1$.
    else
      assign P to the textural model.
      $N_{K-2} \leftarrow N_{K-2} + 1$.
    end if
  else
    $\theta \leftarrow \psi(\arctan \frac{y}{x})$ with $v = (x, y)^T$ and $\psi(a) = a 1_{a \ge 0} + (\pi + a) 1_{a < 0}$.
    find k^* such that $\theta \in [\frac{k^*}{K-2}\pi, \frac{k^*+1}{K-2}\pi)$.
    assign P to the k^*-th model.
    $N_{k^*} \leftarrow N_{k^*} + 1$.
  end if
end while
for $k = 0$ to $K - 1$ **do**
  Set the model prior

$$\boldsymbol{w}_k \leftarrow \frac{N_k}{\sum_{j=0}^{K-1} N_j}.$$

  Set the model mean and covariance

$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|\mathcal{P}_k|} \sum_{P \in \mathcal{P}_k} P, \quad \boldsymbol{\Sigma}_k \leftarrow \frac{1}{|\mathcal{P}_k|} \sum_{P \in \mathcal{P}_k} (P - \boldsymbol{\mu}_k)(P - \boldsymbol{\mu}_k)^T$$

  with $\mathcal{P}_k$ the set of patches attributed to the k-th model.
  Set the factor loading matrix

$$\boldsymbol{F}_k \leftarrow [\ (\lambda_1 - \sigma^2)^{1/2} \phi_1, \cdots, (\lambda_{l_k} - \sigma^2)^{1/2} \phi_{l_k}\ ] \ \text{ with } \ \sigma^2 = \frac{1}{\kappa^2 - l_k} \sum_{m=l_k+1}^{\kappa^2} \lambda_m$$

  where l_k denotes the number of factors required by the k-th model. $V = [\phi_1, \cdots, \phi_{\kappa^2}]$ and $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_{\kappa^2})$ results from the spectral decomposition $\boldsymbol{\Sigma}_k = V \Lambda V^T$.
end for

A few comments on Algorithm 1: first, individual Gaussian factor models are initialized by maximizing the likelihood of N presumably i.i.d.

observations which are not assumed to be noise free

$$(\boldsymbol{F}_*, \sigma_*) = \underset{\boldsymbol{F}\in\mathbb{R}^{\kappa^2\times l},\sigma\in\mathbb{R}_+}{\operatorname{argmax}} \log\prod_{i=1}^{N} \frac{\exp\Big(-\frac{1}{2}(P_i-\mu)^T(\boldsymbol{F}\boldsymbol{F}^T+\sigma^2 I)^{-1}(P_i-\mu)\Big)}{\sqrt{(2\pi)^{\kappa^2}\det(\boldsymbol{F}\boldsymbol{F}^T+\sigma^2 I)}}$$
$$= \underset{\boldsymbol{F}\in\mathbb{R}^{\kappa^2\times l},\sigma\in\mathbb{R}_+}{\operatorname{argmin}} \log\det(\boldsymbol{F}\boldsymbol{F}^T+\sigma^2 I) + \operatorname{tr}\Big((\boldsymbol{F}\boldsymbol{F}^T+\sigma^2 I)^{-1}\Sigma\Big)$$

This problem introduced in Roweis (1998) led to a probabilistic interpretation of the Principal Component Analysis Tipping and Bishop (1999a;b). Its solution turns out to be quite intuitive

$$\boldsymbol{F}_* = [\ (\lambda_1-\sigma^2)^{1/2}\phi_1, \cdots, (\lambda_l-\sigma^2)^{1/2}\phi_l\ ]$$
$$\sigma^2 = \frac{1}{\kappa^2-l}\sum_{m=l+1}^{\kappa^2}\lambda_m$$

with the factor loading $\boldsymbol{F}_*$ unique up to a rotation in its l-dimensional space as the Gaussian distribution $\mathcal{N}(0, I_l)$ is itself rotation-invariant.

Second, the present patch sampling revealed another valuable piece of information of the initial mixture structure, namely $(\boldsymbol{w}_k)_{0\le k\le K-1}$, the prior probability of having a random natural patch belonging to a particular model. It was estimated by

$$\boldsymbol{w}_k = \frac{N_k}{\sum_{j=0}^{K-1} N_j}$$

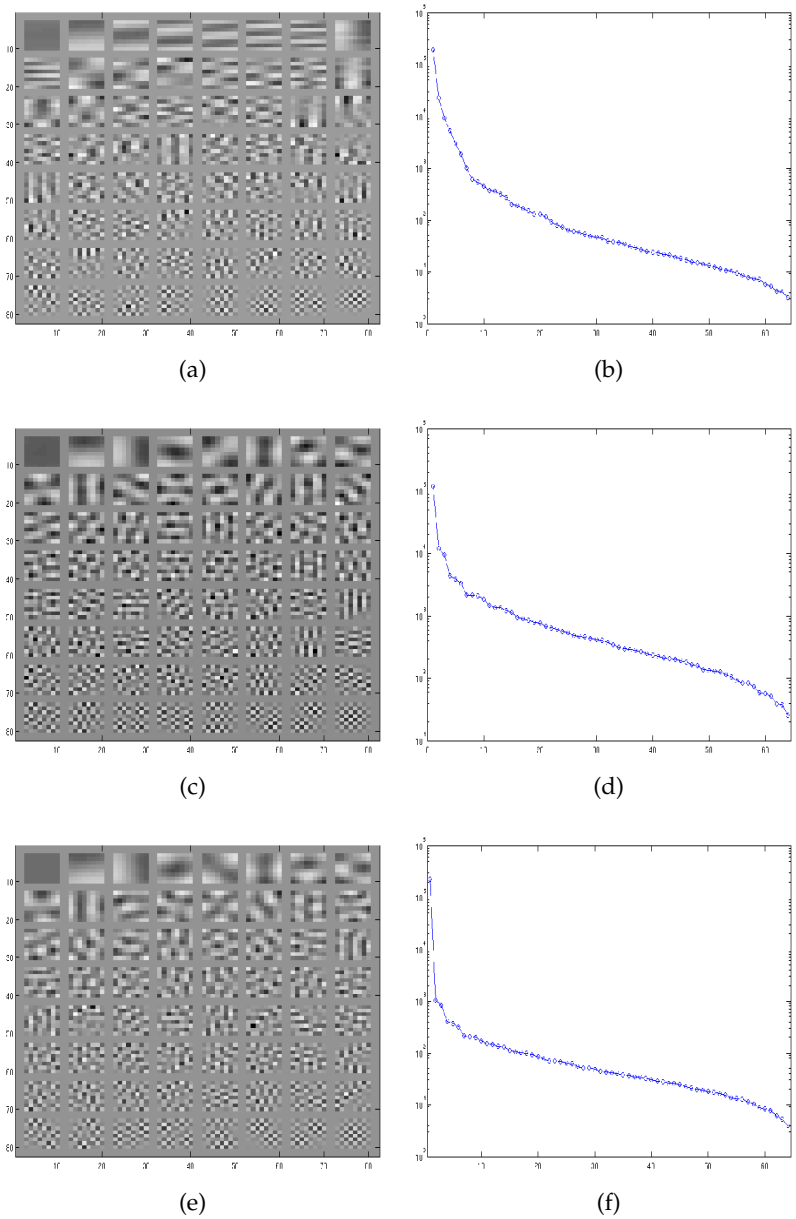
with N_k the number of patches gathered by the k-th model at the end of the sampling. As shown in Fig.1.2(c), the mixing weights are configured in such a way that non-oriented patches are much more likely to appear than their oriented counterparts. Moreover, within the non-oriented category, the flat patches were made to have a slightly higher probability to show up. This setup conveys our belief on the patch composition of a typical natural image and is not image specific: this prior was used in all of our experiments.

In practice, we set $K = 20$ and collected for each model a minimum of 5000 8×8 random patches from the *Berkeley Segmentation Dataset* rendered to grayscale. Shown in Fig.1.1 are three resulting covariance matrices. Several observations are in order:

1. first, leading eigenvectors in the oriented models not only preserve their model feature orientation but also suggest that low frequency patterns tend to appear more often in natural scenes: in view of their much bigger variance, the *true* probability of the natural patch space endowed with the Kolmogorov σ-algebra assigns a predominant weight to low frequency patches;
2. second, the oriented models' eigenvalues do not go to zero as projected by the Gaussian factor models. However, their rapid decay does not deviate far from what is expected of the model either. We kept the first few (e.g. 32) factors and left out the rest;
3. third, contrary to the oriented models, the texture model's eigenvectors resemble those of the flat model. It is the associated eigenvalues

that allows us to tell them apart, a hardly surprising fact given how they are defined. To prevent over-fitting, the first leading eigenvector was made the sole factor representing the flat model, a convenient practice only possible because of the adoption of the Gaussian factor model;

4. finally, to reflect texture's inherent variability, 63 factors are allowed in the textural model. A DCT-like isotropic basis is thus broken up into two to handle two radically different patch categories.



(a) (b) (c) (d) (e) (f)

Figure 1.1 – *examples of the eigenvectors and eigenvalues resulting from Algorithm 1 running on the grayscale Berkeley Segmentation Dataset with* $K = 20$ *and* $\kappa = 8$*: a patch view of the eigenvectors of the (a)* 0*-th (oriented), (c)* 18*-th (textural), (e)* 19*-th (flat) cluster and their eigenvalues displayed in the logarithmic scale: the (b)* 0*-th, (d)* 18*-th and (f)* 19*-th cluster.*

1.2 Orientation based denoising

1.2.1 Orientation based noisy patch classification

To classify patches, we assess their chance of being generated by the individual models in the mixture. For an illustration, we took the image *dice* and added i.i.d. zero-mean Gaussian noise with standard deviation equal to 10. To the three resultant color channels $(u_{\mathbf{R}}, u_{\mathbf{G}}, u_{\mathbf{B}})$, the following PCA inspired transform was applied to enhance the first transformed channel's signal to noise ratio (SNR):

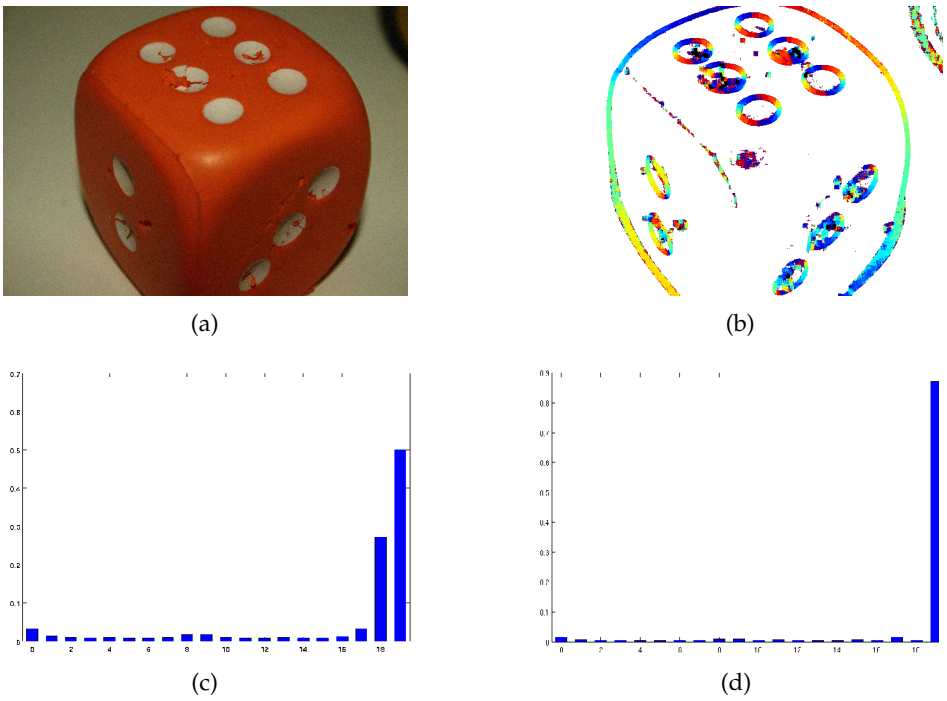
$$\tilde{u}_1 = \frac{u_{\mathbf{R}} + u_{\mathbf{G}} + u_{\mathbf{B}}}{3},\ \tilde{u}_2 = \frac{u_{\mathbf{R}} - u_{\mathbf{B}}}{\sqrt{2}},\ \tilde{u}_3 = \frac{u_{\mathbf{R}} - 2u_{\mathbf{G}} + u_{\mathbf{B}}}{\sqrt{6}} \tag{1.5}$$

To be consistent with the origin (grayscale images) of the calculated statistics, the denominator of the first transform in 1.5 was set to 3 rather than the noise normalizing $\sqrt{3}$.

With the noise standard deviation set to $10/\sqrt{3}$, we ran EM on $\tilde{u}_1$. Each iteration resulted in a set of newly calculated posterior probabilities for every noisy observation $\tilde{P}$ which was used to determine their most likely model with the plug-in version of the Bayes rule Devroye et al. (1996).

$$k_* = \underset{0\le k\le 19}{\operatorname{argmax}}\, \mathbb{P}(s_P = k|\tilde{P}) = \underset{0\le k\le 19}{\operatorname{argmax}}\, \mathbb{P}(\tilde{P}|s_P = k)\mathbb{P}(s_P = k). \tag{1.6}$$

The patch to model mapping, established using 1.6, will be referred to as the *patch map* henceforth. For example, the patch map (Fig.1.2) shows that by the time the first EM iteration ended, pretty much as expected, an overwhelming majority (87.4%) of patches identified with the flat model.



(a) (b) (c) (d)

Figure 1.2 – *(a) noisy image ($\sigma = 10$) (b) patch map formed after one EM iteration (c) initial model priors (d) updated model priors after one EM iteration on the first transformed channel. To visualize the resultant patch map, patches are colorized with white color representing those flat ones.*

1.2.2 SURE aided adaptive filtering

Spatial patch denoising usually projects a noisy patch onto some prefixed basis, it is thus desirable to be able to select bases adaptively so that any given patch can be well represented using just a few coefficients. If this condition proves too difficult to satisfy for every patch, it would be good if it holds true with the patch categories which tend to appear frequently. Our orientation based dictionary construction and selection framework suits this purpose well. For example, if a noisy patch $\tilde{P}$ is found to be best described by the k-th model

$$\tilde{P} = \boldsymbol{F}_k c + \boldsymbol{\mu}_k + \sigma N$$

the eigenvectors of $\boldsymbol{F}_k \boldsymbol{F}_k^T$ constitute a reasonable basis for its representation. It remains to decide how best to filter its resultant coefficients.

We approach the problem with Stein's unbiased risk estimator (SURE) Stein (1981) to construct a statistic indicative of a filter's real time performance. Let us first state a specialized version of the theorem due to Stein in anticipation of its application in this context.

Definition 1.1 *Let $\tilde{P}$ be the sum of a fixed vector $P \in \mathbb{R}^{\kappa^2}$ and a Gaussian random vector composed of i.i.d. entries distributed as $\mathcal{N}(0, \sigma^2)$. Let f be one of the following filters:*

1. *linear: $f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \langle \tilde{P} - \mu, b_j \rangle b_j + \mu$*
2. *soft shrinkage: $f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \gamma_t^{\boldsymbol{soft}}\big(\langle \tilde{P} - \mu, b_j \rangle\big) b_j + \mu$ with $\gamma_t^{\boldsymbol{soft}}(\omega) = sgn(\omega)(|\omega| - t)_+$*
3. *hard shrinkage: $f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \gamma_t^{\boldsymbol{hard}}\big(\langle \tilde{P} - \mu, b_j \rangle\big) b_j + \mu$ with $\gamma_t^{\boldsymbol{hard}}(\omega) = \omega 1_{|\omega| > t}$*

where $\mu, (c_j)_{1 \le j \le \kappa^2}, (b_j)_{1 \le j \le \kappa^2}$, and t are the filter's $\tilde{P}$-independent mean, coefficients, basis, and threshold. And their weak derivatives are defined to be

1. *linear: $\nabla \cdot f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j$*
2. *soft shrinkage: $\nabla \cdot f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j 1_{[t,+\infty)}\big(|\langle \tilde{P} - \mu, b_j \rangle|\big)$*
3. *hard shrinkage: $\nabla \cdot f(\tilde{P}) = \sum_{j=1}^{\kappa^2} c_j \big(1_{[t,+\infty)}\big(|\langle \tilde{P} - \mu, b_j \rangle|\big) + t\mathbb{E}\big[(\delta_t - \delta_{-t})\big(\langle \tilde{P} - \mu, b_j \rangle\big) | P\big]\big)$*

where $\delta_x(\cdot)$ represents a Dirac centered on $x \in \mathbb{R}$.

Theorem 1.1 *Under the assumptions of Definition 1.1, SURE given the observation $\tilde{P}$*

$$\underset{f}{\mathrm{SURE}}(\tilde{P}) := \frac{1}{\kappa^2}\|\tilde{P} - f(\tilde{P})\|^2 - \sigma^2 + \frac{2\sigma^2}{\kappa^2}\nabla \cdot f(\tilde{P})$$

is unbiased

$$\mathbb{E}\big[\underset{f}{\mathrm{SURE}}(\tilde{P}) | P\big] = \mathbb{E}\big[\frac{1}{\kappa^2}\|P - f(\tilde{P})\|^2 | P\big].$$

SURE is valuable in that it depends solely on the observable $\tilde{P}$. However, in case of f being a hard shrinkage operator, the expectation

$$\mathbb{E}\big[(\delta_t - \delta_{-t})\big(\langle \tilde{P} - \mu, b_j\rangle\big)\,|P\big]$$

involves the unknown P. To circumvent the issue, one may replace the expectation with an estimator

$$\frac{1}{2\epsilon}\big(1_{[t-\epsilon,t+\epsilon]} - 1_{[-t-\epsilon,-t+\epsilon]}\big)\big(\langle \tilde{P} - \mu, b_j\rangle\big)$$

for a small $\epsilon > 0$.

Thanks to Theorem 1.1, a useful statistic, the *SURE empirical mean*, can be derived as a measure of how effective filters are. Note that patches in a conventional filtering schema are overlapping, hence correlated. However, given their restricted supports, it is plausible that these patches, seen as a two-dimensional stochastic process, satisfy the wide-sense stationarity Brockwell and Davis (1991), a weaker condition sufficient for the next corollary.

Corollary 1.1 *Under the assumptions of Theorem 1.1 and some mild stationary conditions on image patches $(P_i)_{1\le i\le N}$, the SURE empirical mean*

$$\frac{1}{N}\sum_{i=1}^{N}\underset{f}{\mathrm{SURE}}(\tilde{P}_i) := \frac{1}{N}\sum_{i=1}^{N}\Big(\frac{1}{\kappa^2}\|\tilde{P}_i - f(\tilde{P}_i)\|^2 - \sigma^2 + \frac{2\sigma^2}{\kappa^2}\nabla\cdot f(\tilde{P}_i)\Big) \tag{1.7}$$

is an unbiased estimator of the expected patch MSE $\kappa^{-2}\mathbb{E}[\|P - f(\tilde{P})\|^2]$ and it converges

$$\lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\underset{f}{\mathrm{SURE}}(\tilde{P}_i) = \frac{1}{\kappa^2}\mathbb{E}[\|P - f(\tilde{P})\|^2]$$

almost surely and in $\mathbb{L}^2$.

Sketch of the ideas. Since patches are in abundant supply, the said estimator can be of quite small variance in spite of the terms in the sum being correlated

$$\begin{aligned}
&\frac{1}{N}\sum_{i=1}^{N}\Big(\frac{1}{\kappa^2}\|\tilde{P}_i - f(\tilde{P}_i)\|^2 - \sigma^2 + \frac{2\sigma^2}{\kappa^2}\nabla\cdot f(\tilde{P}_i)\Big)\\
\approx&\mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\Big(\frac{1}{\kappa^2}\|\tilde{P}_i - f(\tilde{P}_i)\|^2 - \sigma^2 + \frac{2\sigma^2}{\kappa^2}\nabla\cdot f(\tilde{P}_i)\Big)\Big] &(1.8)\\
=&\mathbb{E}\Big[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\kappa^2}\|P_i - f(\tilde{P}_i)\|^2\Big] &(1.9)\\
=&\frac{1}{\kappa^2}\mathbb{E}[\|P - f(\tilde{P})\|^2] &(1.10)
\end{aligned}$$

where, under some mild condition on the covariance structure of image patches (see 1.3), the approximation 1.8 is accurate with high probability if N is big enough. Equality (1.9) holds because for all i, we have

$$\mathbb{E}\big[\frac{1}{\kappa^2}\|\tilde{P}_i - f(\tilde{P}_i)\|^2 - \sigma^2 + \frac{2\sigma^2}{\kappa^2}\nabla\cdot f(\tilde{P}_i)\big] = \mathbb{E}\big[\frac{1}{\kappa^2}\|P_i - f(\tilde{P}_i)\|^2\big]$$

thanks to the conditional unbiasedness of SURE. Finally, equality (1.10) stems from $(\|P_i - f(\tilde{P}_i)\|^2)_{1\leq i\leq N}$ sharing the same expectation, one of the defining conditions of the assumed stationarity.

Since the f-dependent estimator 1.7 is a filter performance gauge, we can let both Wiener and shrinkage filters Donoho and Johnstone (1994; 1995) process the noisy patches and then decide the optimal filter for each mixture component based on their respective model-wide SURE empirical mean. The experiments showed that it was better to switch between the hard shrinkage and Wiener filter than sticking with either one of them (see Tab.1.1). Note however that filter selection is not well founded if applied on a patch-by-patch basis because SURE, after all, is a random variable.

Table 1.1 – *SURE Guided Filter Selection (HS for Hard Shrinkage)*

$\sigma = 2$	HS	Wiener	Combined
computer	**1.63**	**1.63**	**1.63**
dice	**0.91**	0.94	**0.91**
flowers	**1.10**	1.23	1.11
girl	1.16	**1.15**	1.16
traffic	**1.71**	1.73	**1.71**
valldemossa	**1.81**	1.87	1.87

$\sigma = 5$	HS	Wiener	Combined
computer	**3.13**	3.20	3.14
dice	**1.34**	1.44	**1.34**
flowers	**1.99**	2.33	**1.99**
girl	**1.79**	1.83	1.80
traffic	3.53	3.56	**3.48**
valldemossa	**3.91**	4.07	4.04

$\sigma = 10$	HS	Wiener	Combined
computer	5.04	5.08	**4.99**
dice	**1.86**	2.01	1.87
flowers	**3.15**	3.50	**3.15**
girl	2.47	2.55	**2.46**
traffic	6.01	5.89	**5.83**
valldemossa	**7.00**	7.08	7.04

$\sigma = 20$	HS	Wiener	Combined
computer	7.84	7.80	**7.73**
dice	**2.64**	2.81	**2.64**
flowers	**4.83**	5.17	4.88
girl	3.47	3.48	**3.45**
traffic	9.43	9.04	**9.01**
valldemossa	11.48	11.36	**11.29**

$\sigma = 30$	HS	Wiener	Combined
computer	10.05	9.91	**9.83**
dice	**3.40**	3.57	3.43
flowers	**6.21**	6.55	6.39
girl	4.26	4.33	**4.24**
traffic	11.69	11.35	**11.29**
valldemossa	14.43	14.26	**14.05**

$\sigma = 40$	HS	Wiener	Combined
computer	11.77	11.71	**11.57**
dice	**4.16**	4.36	4.34
flowers	**7.47**	7.70	7.69
girl	**4.90**	5.11	5.02
traffic	13.40	13.12	**13.02**
valldemossa	16.55	16.48	**16.24**

Perhaps more interestingly, a link can be established between the SURE empirical mean and the recovered image's MSE. To carry out the analysis, we need the next convention:

Definition 1.2 *Let us denote a patch's position by that of its upper-left pixel. A non-overlapping filtering schema including patch (p, q) means a partition of $\mathbb{Z}^2$ consisting of κ-by-κ patches whose coordinates form the set $\{(x, y), (x - p) \bmod \kappa = 0$ and $(y - q) \bmod \kappa = 0\}$.*

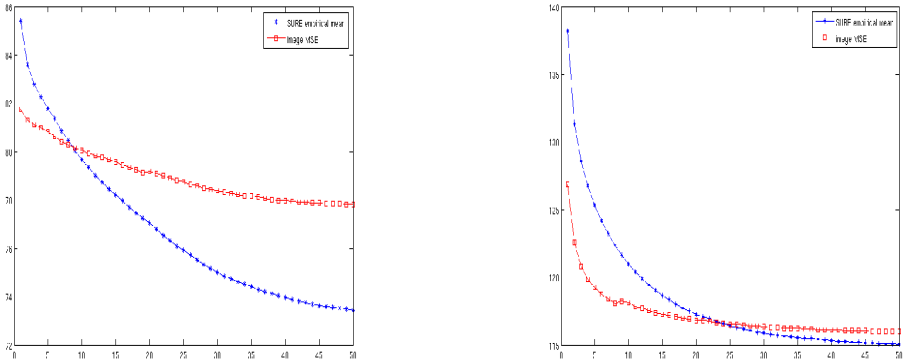
Assume that all κ^2 distinct non-overlapping filtering schemas are used to filter a large $M_U \times N_U$ noisy image $\tilde{U}$ which results in $(\widehat{U}_i)_{1\leq i\leq \kappa^2}$. If we denote U the clean image, Corollary 1.1 implies that for all i the MSE $M_U^{-1}N_U^{-1}\|U - \widehat{U}_i\|^2$ will be close to $\kappa^{-2}\mathbb{E}[\|P - f(\tilde{P})\|^2]$ if both N_U and M_U are big enough. Jensen's inequality leads to

$$\frac{\|\frac{1}{\kappa^2}\sum_{i=1}^{\kappa^2}\widehat{U}_i - U\|^2}{M_U N_U} \leq \frac{\sum_{i=1}^{\kappa^2}\|\widehat{U}_i - U\|^2}{M_U N_U \kappa^2} \approx \frac{1}{\kappa^2}\mathbb{E}[\|P - f(\tilde{P})\|^2]. \qquad (1.11)$$

Asymptotically speaking, the first term in 1.11 is the restoration MSE with the conventional sliding window schema.

Hence the SURE empirical mean becomes a real-time proxy for tracking how well an image gets denoised. Although as an upper bound this

estimator cannot ensure a strict decline of the true MSE, it turned out to be quite reliable in our experiments (Fig. 1.3).



(a) (b)

Figure 1.3 – *the evolution of MSEs and their SURE empirical means as EM iterates on noisy ($\sigma = 20$) (a)* `traffic` *(resp. (b)* `valldemossa`*) in Fig.1.5. These two are quite close. The deviation from the expected asymptotic behavior may be caused by the calculated SURE being biased by the explained approximation in dealing with the patch dependent Wiener filtering coefficients.*

In addition, SURE allows to choose patch size for denoising flat areas. It is hard to overstate the importance of an algorithm's effectiveness at reducing noise in these regions because of the sheer proportion of pixels composing the sky for instance. To help understand the sub-optimality caused by a fixed patch size, consider a slow varying signal $\tau \in \mathbb{Z} \mapsto x_\tau \in \mathbb{R}$ corrupted by some additive white noise n_τ. If the sliding windows of length $\mathfrak{m}$ are applied to recover the signal, its estimated value at time t is

$$\widehat{x}_t = \frac{1}{\mathfrak{m}} \sum_{\tau=t-\mathfrak{m}+1}^{t} \sum_{s=0}^{\mathfrak{m}-1} \frac{x_{\tau+s} + n_{\tau+s}}{\mathfrak{m}} = \frac{1}{\mathfrak{m}^2} \sum_{s=1-\mathfrak{m}}^{\mathfrak{m}-1} \rho_s (x_{t+s} + n_{t+s}) \qquad (1.12)$$

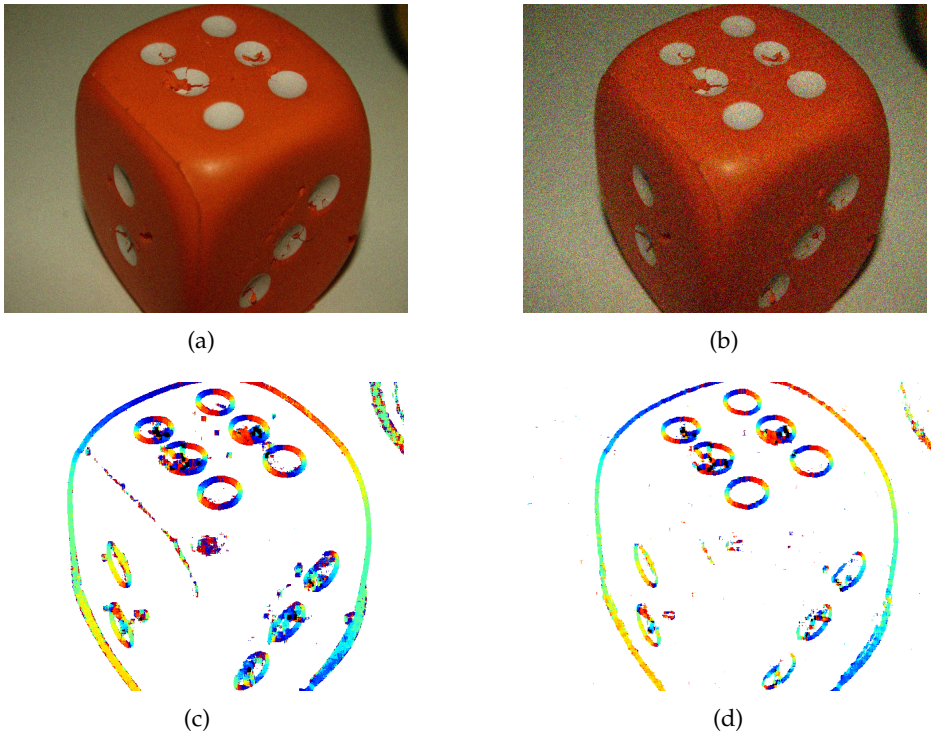
where $\rho_\cdot$ is a probability kernel whose support consists of $2\mathfrak{m} - 1$ elements. If noise is weak, although a bigger patch is conducive to a higher rate of noise reduction, this gain is not significant enough to counterbalance the loss in signal. However, as noise increases in strength, a small patch tends to keep more and more noise which ultimately leads to a poor MSE.

Therefore, we must extend patches whenever necessary. Unlike Buades et al. (2005), Dabov et al. (2007), Lebrun et al. (2013b) which discounts noise correlation of overlapping patches, when denoising a flat patch P, we focus on neighboring patches which do not intersect with P. One such patch Q is deemed similar to P only if the following two conditions hold simultaneously:

1. both patches belong to the same flat region;
2. the true states of P and Q are similar;

The first condition can be checked thanks to the patch map and the connected component labeling algorithm Shapiro and Stockman (2001) while the second one boils down to a chi-square test. Once similar patches are identified, they are merged to form an expanded patch whose estimated intensity is the arithmetic mean of its noisy pixels.

Under strong noise, the rule 1.6 can mistake edges for noise, resulting in some patch orientation not properly recognized, which usually happens in the areas of subtle color transition (Fig.1.4). The chi-square test thus provides a remedy by favoring the locality of patch blending, thereby enhancing the algorithm's robustness.



(a) (b) (c) (d)

Figure 1.4 – *noisy* `dice` *with (a)* $\sigma = 10$ *and (b)* $\sigma = 30$*. EM iterated twice on the first transformed channel as explained in Fig.1.2 to produce the patch map for (c)* $\sigma = 10$ *and (d)* $\sigma = 30$*. Notice that the oriented edge on the top side of the dice failed to be recognized at* $\sigma = 30$*.*

Nonetheless, as stated before, the proposed patch expansion can be problematic at low noise levels. Again, SURE can be our decision aid because the flat areas are restored using simple linear operations similar to those in equation (1.12). The patch expansion can thus be validated or invalidated based on the SURE estimates of the filtered results. Our experiments showed that this automatic device improves dramatically the visual quality as well as the overall MSE of restored images especially when noise is strong.

Algorithm 2 Flat Patch Expansion

Input: patch map $\mathcal{M}$ and noisy patches $(\tilde{P}_i)_{1\leq i\leq N}$.
Parameters: noise variance σ^2, search window size $\mathfrak{w}$ and similarity threshold $\mathfrak{t}$.
Run a connected component labeling algorithm on $\mathcal{M}$ to locate flat areas.
Identify the pixels belonging only to flat patches and put them into a column vector $\tilde{\mathfrak{p}} \in \mathbb{R}^{\mathfrak{n}_f}$.
for $i = 1$ to N **do**
  if $\tilde{P}_i$ belongs to the flat model **then**
    find, in the search window centered on $\tilde{P}_i$, non-overlapping similar patches in the same flat area.
    merge them with $\tilde{P}_i$ to form the expanded noisy patch $\tilde{P}_i^e$.
    estimate all pixels in $\tilde{P}_i^e$ by their arithmetic average which results in $\widehat{P}_i^e$.
    record in a $\mathfrak{n}_f \times \mathfrak{n}_f$ matrix $\mathfrak{F}_i^e$ the filter used in the previous step.
  end if
end for
Restore noisy flat patches with equally weighted patches $\widehat{P}_i^e$. Find the coefficients α_i such that $\mathfrak{F}^e = \sum_i \alpha_i \mathfrak{F}_i$ and $\widehat{\mathfrak{p}}^e = \mathfrak{F}^e \tilde{\mathfrak{p}}$ where $\widehat{\mathfrak{p}}^e$ denotes the restored pixels on the same sites as those in $\tilde{\mathfrak{p}}$.
Calculate the resultant SURE $\mathfrak{S}^e$.
Repeat the same steps without patch expansion and denote the resultant SURE $\mathfrak{S}$.
if $\mathfrak{S}^e < \mathfrak{S}$ **then**
  Take the estimates with patch expansion.
else
  Take the estimates without patch expansion.
end if

1.2.3 Pseudo-code and numerical results

Algorithm 3 details the pseudo-code of the proposed denoising algorithm. For an online demo, please visit `http://demo.ipol.im/demo/52/` Wang (2013b).

Algorithm 3 S-PLE

1: **Input:** a noisy gray image $\tilde{U}$.
2: **Parameter:** noise level σ, number of EM iteration S.
3: Set up the patch orientation Gaussian mixture Θ_0 using Algorithm 1.
4: $\mathfrak{E}_0 \leftarrow (\sigma+1)^2$ where $\mathfrak{E}_0$ denotes the initial SURE empirical mean.
5: Compute $\forall \tilde{P} \in \mathcal{P},\ \forall 0 \leq k \leq 19,\ \mathbb{P}_{\Theta_0}(s_P = k|\tilde{P})$ where $\mathcal{P}$ is the set of all 8×8 noisy patches from $\tilde{U}$.
6: **for** $t = 1$ to S **do**
7: Update model priors:

$$\forall 0 \leq k \leq 19,\ \ \boldsymbol{w}_{k,t} = \frac{1}{|\mathcal{P}|} \sum_{\tilde{P} \in \mathcal{P}} \mathbb{P}_{\Theta_{t-1}}(s_P = k|\tilde{P}).$$

8: Update model means:

$$\forall 0 \leq k \leq 19,\ \ \boldsymbol{\mu}_{k,t} = \frac{\sum_{\tilde{P} \in \mathcal{P}} \tilde{P} \mathbb{P}_{\Theta_{t-1}}(s_P = k|\tilde{P})}{\sum_{\tilde{P} \in \mathcal{P}} \mathbb{P}_{\Theta_{t-1}}(s_P = k|\tilde{P})}.$$

9: Update factor loadings for all k:

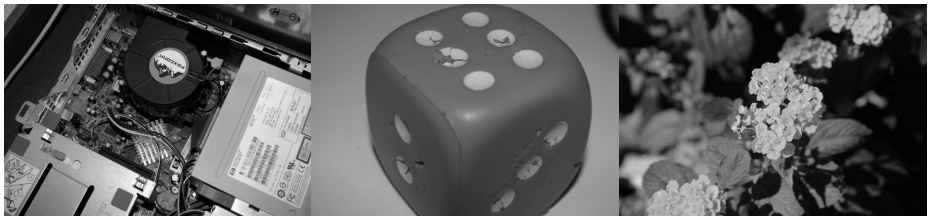
$$\begin{aligned}
\boldsymbol{F}_{k,t} &= \tilde{\boldsymbol{\Sigma}}^*_{k,t-1} \boldsymbol{F}_{k,t-1} (M^{-1}_{k,t-1} \boldsymbol{F}^T_{k,t-1} \tilde{\boldsymbol{\Sigma}}^*_{k,t-1} \boldsymbol{F}_{k,t-1} + \sigma^2 I_{l_k})^{-1} \\
M_{k,t-1} &= \boldsymbol{F}^T_{k,t-1} \boldsymbol{F}_{k,t-1} + \sigma^2 I_{l_k} \\
\tilde{\boldsymbol{\Sigma}}^*_{k,t-1} &= \frac{\sum_{\tilde{P} \in \mathcal{P}} (\tilde{P} - \boldsymbol{\mu}_{k,t})(\tilde{P} - \boldsymbol{\mu}_{k,t})^T \mathbb{P}_{\Theta_{t-1}}(s_P = k|\tilde{P})}{\sum_{\tilde{P} \in \mathcal{P}} \mathbb{P}_{\Theta_{t-1}}(s_P = k|\tilde{P})}
\end{aligned}$$

with $l_k = 32$ for all k except for the last two: $l_{18} = 63$ and $l_{19} = 1$.
10: Create the patch map with the updated parameter set Θ_t:

$$\mathcal{M} : \tilde{P} \in \mathcal{P} \mapsto \underset{0 \leq k \leq 19}{\operatorname{argmax}}\, \mathbb{P}_{\Theta_t}(s_P = k|\tilde{P}).$$

11: For all k, filter the patches assigned to the k-th model with both Wiener and the hard shrinkage filter and pick the better results based on their achieved model-wide SURE empirical mean.
12: Record the SURE empirical mean $\mathfrak{E}_t$.
13: Try patch expansion in flat areas using Algorithm 2.
14: **if** $\mathfrak{E}_t > \mathfrak{E}_{t-1}$ **then**
15: Break (Or continue iterating to see if the SURE empirical mean will eventually go below $\mathfrak{E}_{t-1}$).
16: **end if**
17: **end for**
18: Assign equal weights to all restored patches and recover the image.

Our algorithm (S-PLE) is tested on a grayscale IPOL image set (Fig.1.5) against PLE Yu et al. (2012), Wang (2013b), DCT Yu and Sapiro (2011), GSM Portilla et al. (2003), Rajaei (2014), KSVD Elad and Aharon (2006), Lebrun and Leclaire (2012), NLM Buades et al. (2005; 2011a), EPLL Zoran and Weiss (2011) (source code from `http://www.cs.huji.ac.il/~daniez/`), BM3D Dabov et al. (2007), Lebrun and NL-Bayes Lebrun et al. (2013b;a). The results are compiled in Tab.1.2 where the algorithms are ordered according to their respective performance in RMSE.



(a) (b) (c)

(d) (e) (f)

Figure 1.5 – *test images (a) computer* (704×469) *(b) dice* (704×469) *(c) flowers* (704×469) *(d) girl* (704×469) *(e) traffic* (704×469) *(f) valldemossa* (769×338)

Table 1.2 – *Algorithm Comparison*

$\sigma = 2$	PLE	DCT	GSM	KSVD	NLM	EPLL	S-PLE	BM3D	NL-Bayes
computer	2.40	1.65	1.64	1.55	1.64	1.57	1.54	**1.52**	1.85
dice	0.96	0.91	0.92	0.96	0.97	0.89	0.86	**0.84**	1.31
flowers	1.25	1.08	1.09	1.09	1.29	1.09	**1.02**	1.04	1.44
girl	1.24	1.13	1.12	1.14	1.17	1.09	1.09	**1.05**	1.50
traffic	2.82	1.73	1.77	1.65	1.72	1.64	1.67	**1.62**	1.97
valldemossa	3.65	1.75	1.79	1.73	1.76	1.69	1.78	**1.68**	2.12
average	2.05	1.37	1.38	1.35	1.42	1.32	1.32	**1.29**	1.69

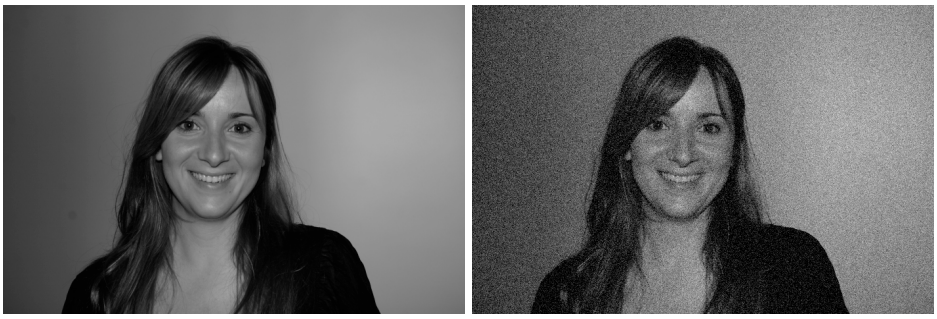
$\sigma = 5$	PLE	DCT	GSM	KSVD	NLM	EPLL	S-PLE	BM3D	NL-Bayes
computer	4.25	3.40	3.28	3.08	3.19	3.05	2.97	**2.94**	2.95
dice	1.45	1.44	1.51	1.89	1.70	1.32	1.29	**1.27**	1.72
flowers	2.16	1.97	1.97	2.11	2.42	1.87	**1.79**	1.81	2.18
girl	1.92	1.85	1.89	2.11	2.01	1.74	**1.69**	**1.69**	1.93
traffic	4.84	3.76	3.69	3.49	3.70	**3.38**	**3.38**	3.40	3.63
valldemossa	6.48	4.04	3.98	3.90	4.15	**3.75**	3.81	3.77	3.85
average	3.51	2.74	2.72	2.76	2.86	2.51	**2.48**	**2.48**	2.71

$\sigma = 10$	PLE	DCT	GSM	KSVD	NLM	EPLL	S-PLE	BM3D	NL-Bayes
computer	6.12	5.66	5.36	5.14	5.16	4.89	4.77	4.65	**4.51**
dice	2.08	2.08	2.24	3.42	2.80	1.90	**1.80**	1.82	2.15
flowers	3.26	3.14	3.19	3.70	4.01	2.92	**2.85**	2.86	3.07
girl	2.65	2.61	2.82	3.60	3.21	2.44	**2.35**	**2.35**	2.56
traffic	7.18	6.51	6.21	5.99	6.05	5.61	5.68	5.67	**5.57**
valldemossa	9.24	7.45	7.04	6.94	7.02	6.58	6.65	6.66	**6.51**
average	5.08	4.57	4.47	4.79	4.70	4.05	**4.00**	**4.00**	4.06

$\sigma = 20$	PLE	DCT	GSM	KSVD	NLM	EPLL	S-PLE	BM3D	NL-Bayes
computer	8.86	8.82	8.37	8.56	7.90	7.54	7.41	7.18	**7.07**
dice	3.20	3.05	3.19	6.74	3.55	2.95	**2.66**	2.67	2.76
flowers	4.97	4.88	5.02	6.60	5.66	4.57	4.55	**4.48**	4.67
girl	3.84	3.65	4.33	6.55	4.18	3.55	3.35	**3.28**	3.40
traffic	10.37	10.08	9.82	9.71	9.40	**8.70**	8.80	8.83	8.74
valldemossa	13.26	12.26	11.55	11.47	11.19	10.60	10.73	10.77	**10.53**
average	7.41	7.12	7.04	8.27	6.98	6.31	6.25	6.20	**6.19**

$\sigma = 30$	PLE	DCT	GSM	KSVD	NLM	EPLL	S-PLE	BM3D	NL-Bayes
computer	10.97	11.13	10.83	10.22	10.43	9.51	9.39	**9.09**	9.12
dice	4.43	3.88	4.18	6.09	4.87	3.94	3.37	3.44	**3.35**
flowers	6.44	6.37	6.15	6.95	7.45	6.00	5.92	**5.80**	5.89
girl	4.85	4.46	4.64	6.24	5.45	4.51	4.11	**4.04**	4.10
traffic	12.23	12.38	12.35	11.58	12.11	**10.85**	11.08	10.97	10.99
valldemossa	15.80	15.32	14.74	14.20	14.37	**13.33**	13.58	13.64	13.43
average	9.12	8.92	8.81	9.21	9.11	8.02	7.90	7.83	**7.81**

$\sigma = 40$	PLE	DCT	GSM	KSVD	NLM	EPLL	S-PLE	BM3D	NL-Bayes
computer	12.61	12.92	12.85	12.20	12.41	11.13	11.24	**10.72**	10.85
dice	5.85	4.64	4.96	7.91	5.20	4.80	4.49	4.14	**3.95**
flowers	7.68	7.59	7.32	8.55	8.96	7.12	7.14	**6.94**	6.98
girl	6.07	5.23	6.01	7.86	5.84	5.30	5.17	4.67	**4.60**
traffic	13.87	14.17	14.70	13.61	14.24	**12.53**	12.86	12.70	12.90
valldemossa	17.71	17.48	17.22	16.52	16.90	**15.57**	15.83	15.73	15.62
average	10.63	10.33	10.51	11.10	10.59	9.40	9.45	**9.15**	**9.15**



(a)

(b)

(c)

(d)

(e)

(f)

Figure 1.6 – *(a) original image (b) noisy image* $\sigma = 20$ *(c) EPLL RMSE* $= 3.55$ *(d) S-PLE RMSE* $= 3.35$ *(e) BM3D RMSE* $= 3.28$ *(f) NL-Bayes RMSE* $= 3.40$

(a) (b)

(c) (d)

(e) (f)

Figure 1.7 – *(a) original image (b) noisy image* $\sigma = 20$ *(c) EPLL RMSE* $= 10.60$ *(d) S-PLE RMSE* $= 10.73$ *(e) BM3D RMSE* $= 10.77$ *(f) NL-Bayes RMSE* $= 10.53$

Though widely accepted, RMSE is not a perfect metric because unlike human eyes, it is insensitive to details. For instance, Fig.1.6(d) and Fig.1.6(f) show that a smoother background makes a more pleasant looking portrait than 1.6(e), notwithstanding a higher RMSE. Similarly, Fig.1.7(d) show that owing to a more subtle approach to identifying flat areas, S-PLE preserves more fine structures than NL-Bayes 1.7(f).

However, 1.7(d) also reveals that because of its intrinsically local approach to orientation detection, structures only observable on a larger scale cannot be kept well. In relation to BM3D and NL-Bayes, a tendency for S-PLE to overlook structures amid strong noise is also documented by the image `computer`. This points to another inherent deficiency of S-PLE: patch similarity is not exploited because it is hard for a Gaussian mixture to model patch position.

1.3 Appendix

Let us denote $\mathfrak{N}_N := \mathbb{Z} \cap [1, N]$ in what follows.

Theorem 1.2 *Let $(X_{t,s})_{(t,s)\in\mathbb{Z}^2}$ be a real-valued stochastic process such that*

$$\forall (t,s),(t',s') \in \mathbb{Z}^2,\ \ \mathbb{E}[X_{t,s}] = \mu,\ \ cov[X_{t,s}, X_{t',s'}] = R_X(t-t', s-s')$$

where μ is a constant and $R_X(\cdot,\cdot)$ satisfies

$$\sum_{(t,s)\in\mathbb{Z}^2} |R_X(t,s)| < +\infty.$$

Then the empirical average

$$A_N := N^{-2} \sum_{(t,s)\in\mathfrak{N}_N^2} X_{t,s}$$

converges to μ both almost surely and in $\mathbb{L}^2$ as N goes to infinity.

Proof: because of

$$\mathbb{E}[\|A_N - \mu\|^2] = N^{-4} \sum_{\substack{(t,s)\in\mathfrak{N}_N^2 \\ (t',s')\in\mathfrak{N}_N^2}} R_X(t-t', s-s'),$$

Lebesgue's dominated convergence theorem implies

$$\lim_{N\to\infty} N^{-2} \sum_{\substack{(t,s)\in\mathfrak{N}_N^2 \\ (t',s')\in\mathfrak{N}_N^2}} R_X(t-t', s-s') = \sum_{(t,s)\in\mathbb{Z}^2} R_X(t,s).$$

The convergence in $\mathbb{L}^2$

$$\lim_{N\to\infty} \mathbb{E}[\|A_N - \mu\|^2] = 0$$

follows from

$$\mathbb{E}[\|A_N - \mu\|^2] \leq N^{-2} \sum_{(t,s)\in\mathbb{Z}^2} |R_X(t,s)|. \tag{1.13}$$

In addition, Markov's inequality, combined with (1.13), leads to

$$\forall c > 0,\ \ \sum_{N=1}^{+\infty} \mathbb{P}(|A_N - \mu| \geq c) < +\infty.$$

The almost sure convergence follows from the Borel-Cantelli lemma. □

The condition

$$\sum_{(t,s)\in\mathbb{Z}^2} |R_X(t,s)| < +\infty$$

requires $R_X(\cdot,\cdot)$ to decay fast enough so that instances of $X_{t,s}$ from two sites far from each other do not behave in sync. In view of the proof, this condition can be weakened to

$$\exists \alpha \in [0,1),\ \limsup_{N\to\infty} N^{-\alpha} \sum_{(t,s)\in\mathfrak{N}_N^2} |R_X(t,s)| < +\infty.$$

E-PLE: un algorithme performant d'inpainting

2

Le mélange de gaussiennes permet de décrire le comportement *a priori* des patches. Dans ce chapitre, nous présentons une analyse probabiliste de l'algorithme *piecewise linear estimation* (PLE) appliqué à l'inpainting et, en utilisant efficacement un mélange gaussien, proposons plusieurs améliorations théoriques et numériques.

2.1 Introduction

2.1.1 Inpainting

Inpainting est une technique d'interpolation qui essaie de ramener une image masquée à son état initial en exploitant au mieux l'information présente dans la partie visible de la même image.

Historiquement, l'un des premiers travaux portant sur le sujet est Masnou and Morel (1998) dans lequel les auteurs, en s'inspirant de la formule de co-aire, proposent de minimiser la courbure de l'image à restaurer à travers une fonctionnelle. Ensuite une variante dans l'espace de fonctions à variation bornée Shen and Chan (2002) est suggérée. Depuis les efforts se multiplient et plusieurs approches telles que les EDPs Bertalmio et al. (2000) et les échantillonnages préférentiels Efros and Leung (1999) sont mises en avant. Cette dernière fait partie d'un paradigme qui repose sur la similarité et se décline en plusieurs algorithmes de renom dont Non-Local Means Buades et al. (2005) et BM3D Dabov et al. (2007).

À l'initiative des travaux comme Elad and Aharon (2006), une autre voie de recherche s'oriente vers la construction d'un dictionnaire redondant permettant d'avoir une représentation parcimonieuse des patches. Il en résulte des avancées récentes Zoran and Weiss (2011), Yu et al. (2012) donnant encore un nouveau point du vue via la modélisation gaussienne.

2.1.2 Estimation linéaire par morceaux

L'idée sur laquelle reposent de nombreux algorithmes de restoration d'image Chatterjee and Milanfar (2009), Yu et al. (2012) vient de l'observation qu'à l'échelle du patch, la caractéristique la plus saillante d'une image naturelle est l'orientation. Cela conduit à modéliser les

patches par une famille de modèles gaussiens décrivant approximativement toute inclinaison d'angle possible, à savoir entre 0 et π. Auxquels s'ajoute un modèle supplémentaire pour rendre compte de la classe de patches contenant essentiellement de la texture.

Pour restaurer une image masquée, il ne reste qu'à associer à tout patch partiellement observé son meilleur modèle dans la famille gaussienne et puis d'appliquer des outils usuels consistant à trouver un candidat optimal qui minimise un terme d'attache aux données tout en tenant compte d'une certaine contrainte de régularité.

Par conséquent, la restoration s'opère patch par patch et le filtre de Wiener est utilisé par défaut, d'où le nom estimation linéaire par morceau, ou *piecewise linear estimation* (PLE) en anglais pour cette catégorie d'algorithmes. Pour simplifier le langage, dans tout ce qui suit, nous désignons par PLE uniquement l'algorithme tel qu'il est décrit dans Yu et al. (2012).

Dans ce chapitre, nous présentons une analyse probabiliste de PLE appliqué à l'inpainting et, en utilisant efficacement un mélange gaussien, proposons une variante significative E-PLE (où E signifie *Enhanced*) de manière à lui apporter plusieurs améliorations théoriques et numériques.

2.2 PLE

2.2.1 Construction et sélection de modèle

Afin de souligner la nouveauté de notre algorithme, nous détaillons comment PLE construit l'ensemble de modèles gaussiens et en choisit un pour un patch donné.

Une gaussienne est complètement caractérisée par son espérance et sa covariance. À l'exception du modèle pour les textures, PLE construit les modèles orientés à partir d'exemples venant de quelques images synthétiques en niveaux de gris dont les gradients à l'intérieur de leurs domaines s'inclinent dans la même direction. Une telle image est créée en convoluant un noyau gaussien avec une image dont les pixels n'ont que deux valeurs distinctes et sont rangés de sorte qu'ils peuvent être séparés selon leur valeur par un segment de droite.

Une décomposition spectrale de la covariance estimée donne une base orthonormale. PLE garde tous ses composants pour le modèle en question même si une matrice de covariance représentant une inclinaison θ n'est pas de plein rang puisqu'aucune combinaison linéaire de ses vecteurs propres ne peut produire l'inclinaison $\theta + \frac{\pi}{2}$.

Le modèle pour les textures est construit avec la base orthonormale DCT et possède les mêmes valeurs propres que les modèles orientés qui, à la différence de E-PLE, sont fixées arbitrairement au début de même que leur espérance.

Dès lors supposons qu'il y a K modèles gaussiens. Soit un patch partiellement observé $\tilde{P}$. PLE le filtre K fois au total en maximisant la vraisemblance sous chacun de ces modèles

$$\widehat{P}_k = \underset{P}{\operatorname{argmin}} \frac{\|\mathfrak{M}P - \tilde{P}\|^2}{\sigma^2} + (P - \mu_k)^T \Sigma_k^{-1} (P - \mu_k) \tag{2.1}$$

où $\mathfrak{M}$, σ, μ_k, Σ_k désignent le masque associé au patch $\tilde{P}$, le niveau de bruit blanc des pixels visibles, l'espérance et la covariance du modèle k pour l'indice k allant de 1 à K.

La sélection du modèle se fait ensuite par l'évaluation des probabilités conditionnelles

$$k_* = \underset{1 \leq k \leq K}{\operatorname{argmax}}\, p(\widehat{P}_k, \tilde{P} \mid \mu_k, \Sigma_k) \tag{2.2}$$

avec $\widehat{P}_{k_*}$ retenu comme le meilleur estimateur de l'état initial.

Enfin, nous mettons à jour tous les modèles avec les estimateurs qui leurs sont attribués et entamons un nouvel cycle de filtrage et d'estimation si besoin.

2.2.2 Comment améliorer PLE ?

Malgré ses bons résultats, PLE reste améliorable.

Premièrement, si l'algorithme décrit précédemment est appelé EM par ses auteurs, il ne s'agit pas à proprement parler de *expectation maximization* Dempster et al. (1977), la classe d'algorithmes reconnue pour leur capacité de faire croître la vraisemblance d'un mélange sur lequel PLE ne repose pas car son modèle manque la probabilité *a priori* w.

$$p(P) = \sum_{k=1}^{N} w_k \mathcal{N}(P \mid \mu_k, \Sigma_k) \tag{2.3}$$

qui ne peut être inférée par l'échantillonnage synthétique.

Deuxièmement, il est préférable que les modèles soient alimentés par de vraies données dès le départ, ce qui permet d'accélérer la phase d'apprentissage de l'algorithme face aux images réelles.

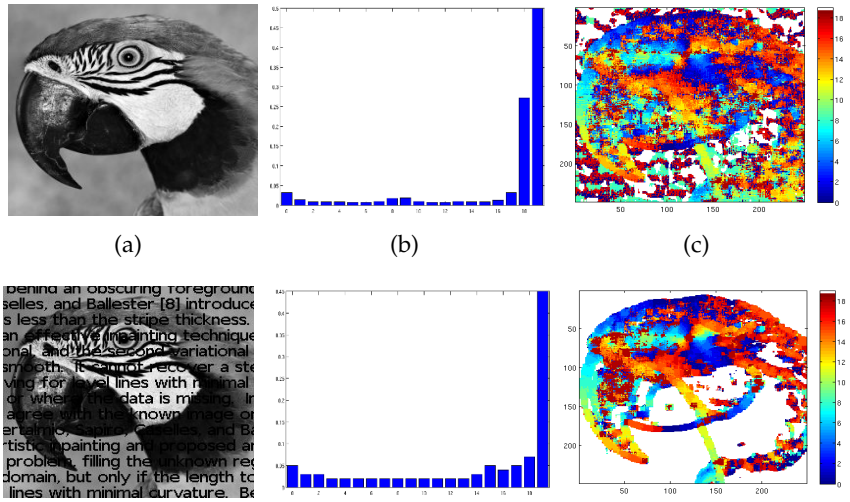
Finalement, comme les covariances des patches orientés n'ont pas besoin d'être de plein rang, restreindre le nombre de leurs vecteurs propres réduit le risque de surapprentissage et aussi le coût de calcul.

2.3 E-PLE

2.3.1 La classification des patches avec EM

La construction du mélange suit la procédure expliquée dans 1.1. Similaire à 1, l'algorithme EM Dempster et al. (1977) est adapté à notre cadre où les données ne sont que partiellement observées afin d'inférer les paramètres du mélange, y compris ceux pour caractériser les modèles individuels et leur poids probabiliste w_k. La classification des patches vise à miniser le risque de Bayes 1.6

La figure 2.1 illustre un exemple de classification des patches. L'échantillonnage nous donne une initialisation raisonnable du mélange composé de 20 modèles À l'issue de trois itérations d'EM, les modèles orientés se voient attribuer plus d'importance au détriment du modèle texturé, un comportement auquel nous nous attendons bien.



(a) (b) (c)

(d) (e) (f)

Figure 2.1 – *La classification des patches masqués à l'aide d'EM. (a) image originale (d) image masquée (b) la probabilité a priori des modèles initialisés (e) la probabilité à priori inférée par EM au bout de sa troisième itération (c) (resp. (f)) carte d'association patch-modèle au bout de la première (resp. troisième) itération où les couleurs représentent les différents modèles dont la couleur blanche pour le modèle plat.*

2.3.2 Résultats numériques et discussions

Dans cette étude numérique, nous avons pris quatre images et deux types de masques, à savoir le masque à texte et le masque aléatoire où tout pixel a une certaine probabilité d'être rendu invisible. Les algorithmes sont itérés six fois dans tous les cas. Le critère d'évaluation est RMSE (*root mean square error*).

Les résultats montrés au tableau 2.1 confirment que plus les images à traiter sont dégradées, plus l'effet dû à l'incorporation de la probabilité de mélange se fait sentir car c'est à ce moment là que nous avons le plus besoin de faire appel à la connaissance *a priori* pour déterminer les appartenances des patches.

En pratique, avec moins de facteurs à calculer, E-PLE est beaucoup plus rapide que PLE, d'autant plus que grâce à une initialisation plus adaptée, souvent E-PLE n'a besoin que d'une seule itération pour parvenir à un bon résultat en RMSE tandis que PLE typiquement en nécessite beaucoup plus.

(a) (b) (c)

Figure 2.2 – *D'autres images utilisées dans la comparaison des algorithmes. (b) formes (c) barbara (d) grenouille*

Table 2.1 – *Comparison des algorithmes*

text	barbara	grenouille	parrot	formes
PLE	**4.2**	**4.3**	**6.1**	5.9
E-PLE	5.0	4.7	6.6	**5.7**

rand 0.2	barbara	grenouille	parrot	formes
PLE	**1.7**	**2.0**	**3.7**	2.4
E-PLE	1.9	2.2	3.8	**2.3**

rand 0.4	barbara	grenouille	parrot	formes
PLE	**3.7**	**4.0**	6.9	5.4
E-PLE	**3.7**	4.1	**6.7**	**4.7**

rand 0.6	barbara	grenouille	parrot	formes
PLE	10.6	7.1	10.9	11.2
E-PLE	**7.9**	**6.8**	**10.0**	**8.8**

rand 0.8	barbara	grenouille	parrot	formes
PLE	20.1	11.0	16.0	19.4
E-PLE	**16.9**	**10.8**	**14.8**	**16.5**

Part II. Conditional expectation and neural networks

The second part focuses on multilayer neural networks and their applications to image processing. The fundamental observation is that driven by backpropagation, feedforward neural networks seek to approximate a conditional expectation. Though larger neural networks tend to perform better, the following work mainly explores the possibility to reduce their sizes and training cost.

We have chosen image demosaicking and denoising because unlike random masks arising in the context of image inpainting, Bayer color filter array and additive Gaussian noise are two types of distortion with locally defined data format and thus fit well into the patch regression framework.

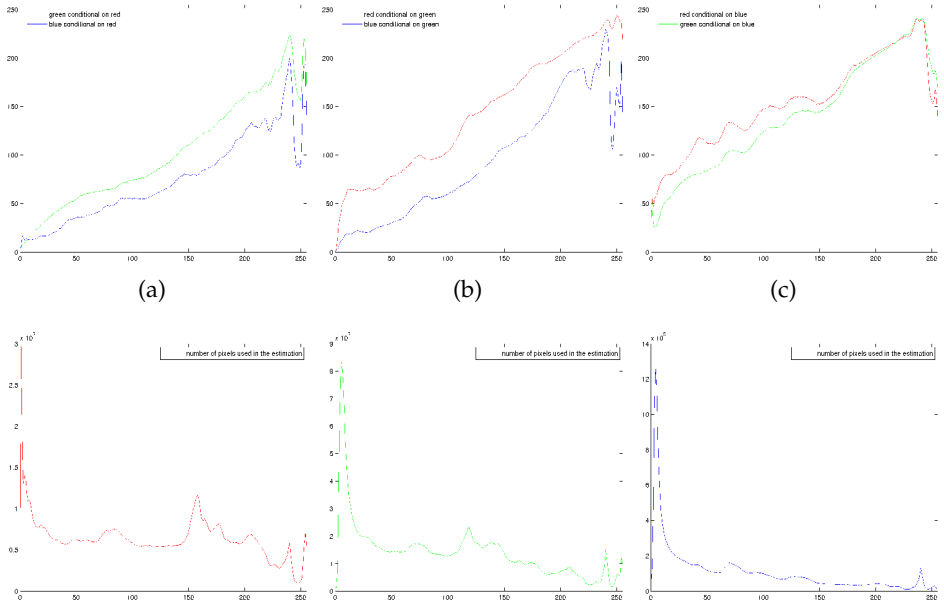
3 A multilayer neural network for image demosaicking

The recent revival of interest in artificial neural networks has been fueled by their successful applications in various image processing and computer vision tasks. In this chapter, we make use of the rotational invariance of the natural image patch distribution and propose a multilayer neural network for demosaicking 4×4 image patches. We show that it does surprisingly well compared to state-of-the-art approaches requiring much larger input sizes.

3.1 Introduction

Most digital cameras place a Bayer Bayer (1976) color filter array (CFA) in front of the sensor to record just one color (R, G or B) at each pixel site. This results in a *mosaiced* image. The process of restoring the missing color information in such an image through cross-channel interpolation is termed *demosaicking*. A huge body of work has been done in this field Li et al. (2008) including two neural network inspired approaches Kapah and Hel-Or (2000), Go et al. (2000). However, recent successful applications of multilayer neural networks in image processing Burger et al. (2012) warrant a revisit of the subject.

Evaluated in MSE, a patch based demosaicking algorithm should approximate the expectation of the missing pixels conditional on the visible ones under some patch distribution. For an illustration, we start with pixels, or 1×1 patches, for which conditional expectations are simple to evaluate. For instance, to estimate the green channel given its red counterpart, it suffices to sample a large number of color pixels having the same red intensity and take the average of their green intensities (see Fig.3.1). Unsurprisingly, despite their accuracy, the obtained conditional expectations do not work well even when applied to an image taken from the same image set (see Fig.3.2). Because of the quasi-linear relationship in all three cases, here demosaicking tends to make a gray-looking image out of a mosaiced one. In addition, since the local geometry is not taken into account, the so-called zipper effect is pronounced.



(a) (b) (c)

(d) (e) (f)

Figure 3.1 – *In the McMaster dataset Zhang et al. (2011): conditional expectations of the other two channels given the (a) red (b) green (c) blue channel and their respective un-normalized prior distributions (d) red (e) green (f) blue.*

(a) (b)

Figure 3.2 – *Demosaicking a McMaster subimage with pixel-wise conditional expectations computed from the same dataset.*

Therefore, the natural extension is to enlarge patches. Its benefit is obvious from the previous discussion, even though computing conditional expectations are no longer as easy, because the same approach quickly becomes intractable due to the curse of dimensionality. Fortunately, studies have shown that neural networks are universal approximators Barron (1994), Hornik et al. (1989) and that deep networks tend to represent signals more efficiently Bengio (2009). Owing to the problem's non-convex nature in general, it was not clear as to how best to tap into this potential from a numerical standpoint. However, since 2006, major advances have been made in this direction Bengio (2009). One way to train a deep, or mul-

tilayer, neural network is an autoencoder-enabled unsupervised learning process.

3.2 Training a deep neural network

First, let us introduce the concept of autoencoder. An autoencoder is a neural network whose fan-in x and fan-out $\widehat{x}$ are vectors of the same dimension, related by a non-linear hidden layer of potentially different size

$$\begin{aligned} \widehat{x} &= Va + h \\ a &= \tanh(Wx + b) \end{aligned} \tag{3.1}$$

or

$$\begin{aligned} \widehat{x} &= \tanh(Va + h) \\ a &= \tanh(Wx + b) \end{aligned} \tag{3.2}$$

where $\tanh(\cdot)$ is understood to be applied element-wise. The filtering defined by (W, b) followed by the hidden layer transforms the fan-in x to the activation a, which is supposed to code x in such a way as to make the reconstruction $\widehat{x}$ close to x. Therefore, building an autoencoder can be seen as extracting the most statistically relevant features from the fan-in. Suppose that we have n input signals $(x_j)_{1\leq j\leq n} \in \mathbb{R}^{d\times n}$ which produce n activations $\mathfrak{a} := (a_j)_{1\leq j\leq n} \in \mathbb{R}^{\mathfrak{m}d\times n}$, the autoencoder's parameters are determined through minimizing the following objective

$$\frac{1}{nd}\sum_{j=1}^{n}\|\widehat{x}_j - x_j\|_2^2 + \frac{\alpha}{\mathfrak{m}d^2}(\|V\|_2^2 + \|W\|_2^2) + \beta\mathfrak{sp}(\mathfrak{a}, n, d, \rho) \tag{3.3}$$

with α, β, ρ three prefixed hyper-parameters. The last penalty, intended to induce an over-complete dictionary for sparse representation, is a function of average activation

$$\mathfrak{sp}(a_1, \cdots, a_n, n, d, \rho) = \frac{1}{\mathfrak{m}d}\sum_{k=1}^{\mathfrak{m}d} KL\big(\rho\big|\big|\frac{1}{n}\sum_{j=1}^{n}|a_{jk}|\big)$$

where

1. $\mathfrak{m}$ is a redundancy control. Together with the activation level $\rho \in (0, 1)$, it suggests that ideally, $\rho\mathfrak{m}d$ hidden nodes per patch are fully activated on average.
2. $KL(\cdot||\cdot)$ is the standard Kullback-Leibler divergence between two Bernoulli probabilities. Regardless of ρ, this divergence strongly penalises average activation near 0 and 1, either because the associated feature is irrelevant or because it has rather low information value. Yet it is still possible for a spurious feature to reach the prescribed activation level if its norm is big enough, hence the regularization term $\|W\|_2^2$.

3. The term $\|V\|_2^2$ is there to prevent over-fitting. Another concern is to prevent an autoencoder from learning an identity, which may come with a small W and a large V to exploit the quasi-linearity of $\tanh(\cdot)$ around zero. Thus the prior on V helps. See Vincent et al. (2008) for another solution.
4. The features recorded in W ought to stay away from 0 thanks to the sparsity constraint. If there are not enough distinct features to fill all the hidden nodes, almost identical features may result, which indicates a higher than necessary $\mathfrak{m}$ or poorly chosen hyper-parameters.

Turn now to our deep neural network. Given a $\kappa \times \kappa$ mosaiced patch v (hence $d = \kappa^2$), represented as a $\mathbb{R}^{\kappa^2}$-valued column vector, our network with two hidden and one linear output layers will make an educated guess $\widehat{u}$ as to what the missing pixels $u \in \mathbb{R}^{2\kappa^2}$ should look like according to

$$\widehat{u} = W_3[v^t, \tanh(W_2 \tanh(W_1 v + b_1) + b_2)^t]^t + b_3.$$

Note that this architecture is meant to let the neural network focus on the non-linear part of the color interpolation.

To determine $(W_l, b_l)_{1 \leq l \leq 3}$, a combination of supervised and unsupervised learning, as advocated in Bengio (2009), was used. We first drew n random patches from some high quality color images. To each of these patches, the RGGB Bayer CFA was applied to separate the visible pixels v from the invisible ones u. An autoencoder was trained on v and the resultant filter (W, b) henceforth became (W_1, b_1). The activations from the first hidden layer were then computed, on which another autoencoder was trained to set up (W_2, b_2). Finally, the output linear layer was initialized by a linear regression to fit the teaching signals u. All the involved objectives (3.3) were optimized using L-BFGS Nocedal (1980). Note that before starting training, it helps to zero-phase whiten v and u (see Algorithm).

Zero-phase whitening

1: **Input:** n vectors $p_j \in \mathbb{R}^d$.
2: **Output:** the whitening operator $\mathcal{W}$.
3: **Parameter:** smoothing factor σ_Z.
4: Center the inputs $\tilde{p}_j = (p_j - \bar{p})/255$ with $\bar{p} = \frac{1}{n}\sum_{j=1}^n p_j$.
5: Run the principal component analysis (PCA) on $(\tilde{p}_j)_{1 \leq j \leq n}$ to get their eigenvectors and eigenvalues $(\phi_i, \lambda_i)_{1 \leq i \leq d}$.
6: $\mathcal{W}(p) = 255^{-1} \sum_{i=1}^d \sqrt{\frac{1}{\lambda_i + \sigma_z}} \langle \phi_i, p - \tilde{p} \rangle \phi_i$ with $\langle \cdot, \cdot \rangle$ a standard scalar product.

Once fully configured, the deep network was fine-tuned with the back-propagation algorithm LeCun et al. (1998) on the same training data for several more rounds. Fresh data was then drawn in to run the stochastic gradient in the hope of further improving the network. In these last two stages, the network's generalization error was assessed on a distinct set of validation examples. Since we could not know in advance the hyper-parameters most conducive to good learning, an exhaustive exploration in the parameter space was carried out.

Trained to restore color removed by the RGGB CFA represented in Fig.3.3(a) for 2×2 patches, a network should be able to handle those masked by any of the other three CFAs as well because the patch distribution is rotation-invariant. The same argument remains true when extended

to $2k \times 2k$ patches with $k \geq 1$. In practice, it means that each missing color pixel can be estimated multiple times, thereby enhancing the algorithm's overall performance.

(a) (b) (c) (d)

Figure 3.3 – *Four basic blocks of the Bayer pattern*

3.3 Experiments

In our experiments, the patches were 4×4 and whitened using Algorithm with $\sigma_Z = 0.1$ for inputs and $\sigma_Z = 0$ for outputs. The first hidden layer containing 80 nodes was trained 400 rounds using an autoencoder of type (3.1) with $\alpha = 0.4$, $\beta = 5$, $\rho = 0.1$ on 10^5 examples. The second hidden layer also has 80 nodes, and was trained 400 rounds using an autoencoder of type (3.2) with hyper-parameters $\alpha = 0.4$, $\beta = 0.1$, $\rho = 0.05$. The final layer followed from

$$(W_3, b_3) = \underset{(W,b)}{\operatorname{argmin}} \|\tilde{u} - W\bar{v} - b\|_2^2 + \lambda\|W\|_2^2$$

where $\bar{v}$ is the concatenation of the whitened input $\tilde{v}$ and their activations from the network's second hidden layer. $\tilde{u}$ is the whitened output. The decay parameter λ was set to 0.005.

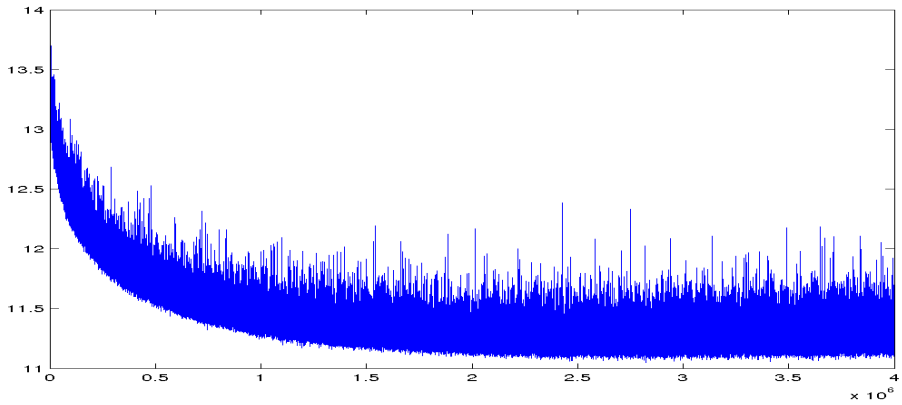
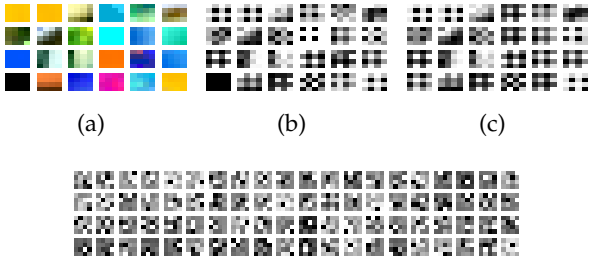


Figure 3.4 – *Evolution of the validation RMSEs during the stochastic gradient descent. Though noisy, this RMSE path clearly demonstrates the value of stochastic gradient descent.*

When tracking the neural network's performance (Fig.3.4) on a validation dataset of 10^4 patches, we ran 4×10^6 rounds of stochastic gradient descent with batch size 100 at a learning rate 0.001 to further drive down the objective

$$\|\tilde{u} - f(\tilde{v}, \theta)\|_2^2 + \lambda\|W_3\|_2^2,$$

where $f(\cdot, \theta)$ is the network defined by its parameter set θ.



(a) (b) (c)

(d)

Figure 3.5 – *Examples of 3.5(a) color patches 3.5(b) RGGB CFA filtered patches and 3.5(c) zero-phase whitened patches used in the training. 3.5(d) All the 80 learned features stored in the first hidden layer of the trained network.*

The training and validation examples were collected from two distinct sets, having respectively 2992 and 10 *Flickr* images. Having been demosaicked one way or another, these images were not fed directly to the neural network or it might wind up simply imitating other demosaicking algorithms. To guarantee their quality, we downsampled the images by 2 after convolving them with a Gaussian kernel of standard deviation equal to 1.2 Morel and Yu (2011).

We tested our neural network against some best performing methods available. The results, stated in both RMSE and the zipper effect as in Buades et al. (2009), are reported in Tab.3.1 for the two datasets (Fig.3.6). As stated in Mairal et al. (2009), the last five *Kodak* images were used to tune KSVD. They were excluded for fairness.

Figure 3.6 – *The Kodak and McMaster dataset*

3.4 Discussion

The experiments show that our tiny neural network compares favorably with these state-of-the-art algorithms. We have deliberately chosen images of vibrant colors to constitute our training set, which explains in part that the resultant network works better on *McMaster* than on *Kodak*. Fig.3.7 shows that the network handles abrupt color transitions remark-

McMaster	**NN**	**KSVD**	**CS**	**SSD**	**HA**	**Paliy**	**ZW**
1	**8.52**	9.67	8.92	10.60	10.45	11.61	11.65
2	**4.66**	4.89	5.03	5.03	5.08	5.36	5.44
3	5.75	**5.54**	5.57	5.81	6.53	5.97	6.03
4	3.70	**3.49**	4.37	3.90	4.53	4.48	5.09
5	**4.84**	5.98	5.37	6.17	6.00	6.82	7.20
6	3.61	**3.36**	3.82	4.33	4.41	5.14	5.65
7	3.83	2.68	**2.50**	3.80	4.37	2.78	2.79
8	3.16	**2.96**	3.26	3.46	3.76	3.23	3.49
9	**3.57**	3.65	3.97	4.11	4.32	4.79	5.17
10	**3.05**	3.13	3.25	3.30	3.50	3.88	4.02
11	2.78	**2.74**	2.97	3.06	3.33	3.61	3.66
12	**2.96**	3.20	3.32	3.33	3.54	3.82	3.89
13	**2.42**	2.51	2.61	2.50	2.65	2.99	3.03
14	**2.96**	3.09	3.28	3.15	3.27	3.76	3.60
15	**2.89**	3.02	3.10	3.15	3.29	3.51	3.63
16	**4.82**	6.32	6.15	7.15	6.94	8.92	8.11
17	**5.44**	6.05	5.55	7.40	7.28	8.83	9.10
18	**4.38**	4.71	5.03	5.16	5.22	5.34	5.34
rmse avg.	**4.07**	4.27	4.33	4.74	4.91	5.26	5.38
zipper avg.	0.34	0.34	**0.30**	0.33	0.38	0.38	0.39

Kodak	**KSVD**	**Paliy**	**ZW**	**NN**	**SSD**	**CS**	**HA**
01	**2.23**	2.44	2.73	4.62	4.70	4.09	5.35
02	**2.19**	2.26	2.42	3.32	3.91	4.32	3.49
03	**1.54**	1.69	1.86	2.57	3.05	3.88	3.04
04	**1.98**	2.34	2.53	3.05	3.10	4.05	3.30
05	**2.64**	3.33	3.25	4.19	4.36	4.72	4.88
06	**2.13**	2.30	2.38	4.24	4.16	4.29	4.78
07	**1.64**	1.82	2.06	2.45	2.94	3.82	2.84
08	**3.33**	3.51	3.72	5.72	5.19	5.14	6.32
09	**1.62**	1.67	1.74	2.37	2.64	3.69	2.75
10	**1.78**	1.87	1.92	2.40	2.75	3.71	2.88
11	**2.16**	2.36	2.43	3.80	3.71	4.11	4.09
12	**1.48**	1.62	1.70	2.48	2.57	3.50	2.58
13	**3.74**	3.95	4.19	7.76	6.48	5.10	8.09
14	**2.81**	3.51	3.65	4.01	4.71	5.01	4.66
15	**2.25**	2.62	2.73	3.17	3.94	4.59	3.97
16	**1.44**	1.58	1.56	2.98	3.57	3.95	3.63
17	**1.95**	2.02	2.02	2.95	2.74	2.20	2.97
18	**3.17**	3.65	3.50	5.20	4.55	4.15	5.20
19	**1.95**	2.20	2.19	3.52	3.02	2.54	3.40
rmse avg.	**2.21**	2.46	2.55	3.72	3.79	4.04	4.11
zipper avg.	**0.22**	0.24	0.25	0.36	0.32	0.23	0.35

Table 3.1 – *The results from our neural network (NN), (the Kodak-tuned) KSVD Mairal et al. (2009), CS Getreuer (2011a; 2012), SSD Buades et al. (2009; 2011b), HA Hamilton and Adams (1997), Paliy Paliy et al. (2007) and ZW Zhang and Wu (2005), Getreuer (2011b) on McMaster (top) and Kodak (bottom). The row-wise best results are in bold. The columns are ordered to reflect the performances in RMSE. We only show the average zipper effects for lack of space.*

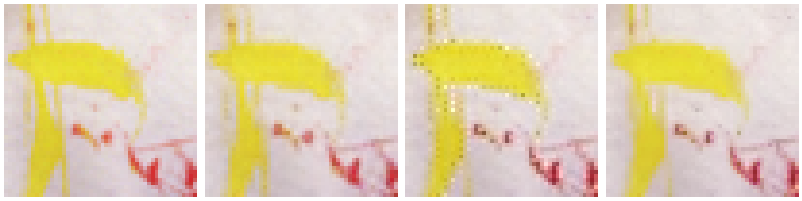
NN	HA	CS	KSVD	ZW	Paliy	SSD
4×4	5×5	5×5	8×8	9×9	12×12	15×15

Table 3.2 – *Comparison of the neighborhood size demanded by the seven algorithms. Note that because of their design, these algorithms do not necessarily use every pixel in a given neighborhood. Nonetheless, the neighborhood size remains a basic indicator of the associated algorithmic complexity.*

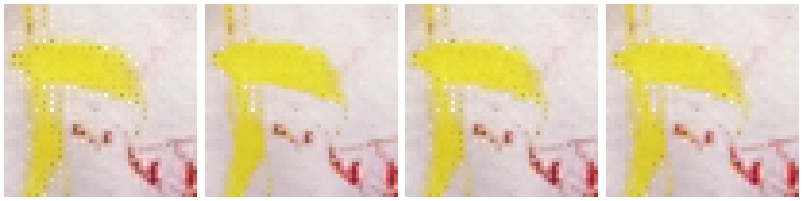
ably well. However, its also reveals the neural network's inability to recover high frequency patterns (Tab.3.2).

Recent evidence Burger et al. (2012) suggests that endowed with a higher capacity, a deep network should perform substantially better with larger patches, because more structural information can then be discovered. However, training a larger network incurs a higher computational cost. It took roughly three days on a 8-core Linux machine to train the current network.

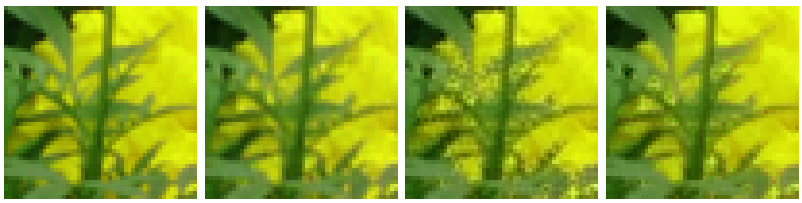
As opposed to many noise sensitive algorithms, the neural network approach may be extended to noisy image demosaicking, which is certainly of a bigger practical interest because denoising and demosaicking could then be handled in one single procedure Hirakawa and Parks (2006), Chatterjee et al. (2011).



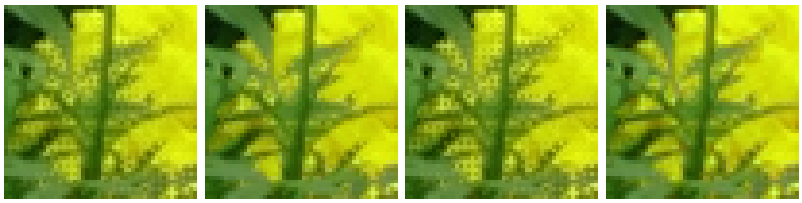
(a) original (b) NN (ours) (c) KSVD Mairal et al. (2009) (d) CS Getreuer (2011a)



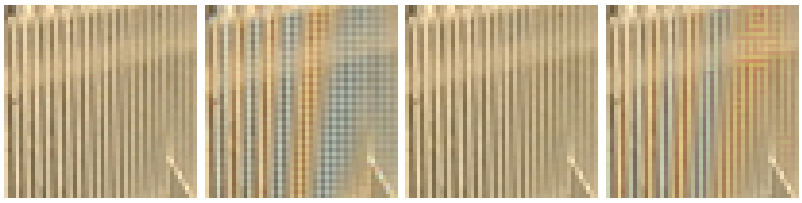
(e) ZW Zhang and Wu (2005) (f) SSD Buades et al. (2009) (g) Paliy Paliy et al. (2007) (h) HA Hamilton and Adams (1997)



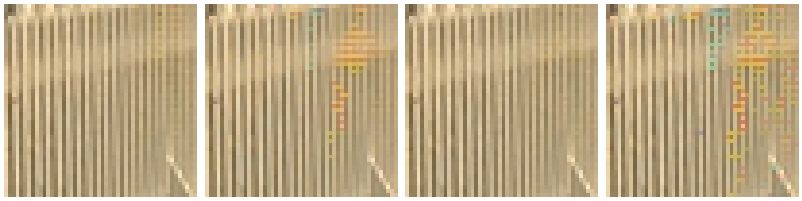
(i) original (j) NN (k) KSVD (l) CS



(m) ZW (n) SSD (o) Paliy (p) HA



(q) original (r) NN (s) KSVD (t) CS



(u) ZW (v) SSD (w) Paliy (x) HA

Figure 3.7 – *Comparison of the seven algorithms: our neural network does not do well when image patterns to recover oscillate much. In contrast, it does interpolate well where color changes abruptly. Visually speaking, contour stencil Getreuer (2011a) yields the closest numerical results. However, the neural network seems to be more accurate on blobs.*

4 Can a single image denoising neural network handle all levels of Gaussian noise?

A recently introduced set of deep neural networks designed for the image denoising task achieves state-of-the-art performance. However, they are specialized networks in that each of them can handle just one noise level fixed in their respective training process. In this chapter, by investigating the image patch distribution invariance with respect to linear transforms, we show how to make a single existing deep neural network work well across all levels of Gaussian noise, thereby allowing to significantly reduce the training time for a general-purpose neural network powered denoising algorithm.

4.1 Introduction

Recently, a set of deep neural networks, or multi-layer perceptrons (MLPs) designed for the image denoising task Burger et al. (2012) has been shown to outperform BM3D Dabov et al. (2007), widely accepted as the state-of-the-art. Although these enormous deep networks only work at the noise levels which they were trained for, this turn of events clearly demonstrates the potential of a pure learning strategy. A philosophical difference sets the two patch-based methods apart: BM3D, a major spin-off of the original non-local means Buades et al. (2005), seeks information exclusively inside the noisy image while the neural network derives all its power by looking at noisy and clean patch pairs gathered from other images. Other algorithms exist Elad and Aharon (2006), Mairal et al. (2009), Zoran and Weiss (2011), Burger et al. (2013) which fall between these two ends of the spectrum.

As pointed out in Chapter 3, whatever their architectures, neural networks as a class of universal approximators Barron (1994), Hornik et al. (1989) seek to approximate the conditional expectation under some input distribution. In our setting, let x, $\tilde{x}$, and y denote the clean, noisy and denoised patch. Note that y does not necessarily have the same dimension as $\tilde{x}$. All the neural networks Burger et al. (2012) under this study, for instance, produce a $y \in \mathbb{R}^{17\times 17}$ based on a noisy observation $\tilde{x} \in \mathbb{R}^{39\times 39}$

which includes not only y's corresponding noisy pixels but also its surrounding ones. Also note that with the *true* natural patch distribution beyond reach, a huge number of patches are drawn stochastically Bottou (2010) from a large natural image dataset to provide the underlying distribution for computing

$$\begin{aligned}\theta^* &= \operatorname*{argmin}_\theta \mathbb{E}\|f(\tilde{x},\theta) - x\|_2^2 \\ &= \operatorname*{argmin}_\theta \mathbb{E}\|f(\tilde{x},\theta) - \mathbb{E}[x|\tilde{x}]\|_2^2\end{aligned} \tag{4.1}$$

where $f(\cdot,\theta) : \tilde{x} \mapsto y$ is a neural network parametrized by its connection weights θ. Due to this problem's non-convex nature in general, instead of the potentially intractable θ^*, a good θ, judged on the basis of the resulting network's generalization error, is usually accepted as a solution.

In spite of their impressive performance, the proposed deep networks are impractical. As stated in the paper itself Burger et al. (2012): *our most competitive MLP is tailored to a single level of noise and does not generalize well to other noise levels. This is a serious limitation which we already tried to overcome with an MLP trained on several noise levels. However, the latter does not yet achieve the same performance for* $\sigma = 25$ *as the specialized MLP*. Since a standard general-purpose algorithm for Gaussian noise removal ought to be able to handle all levels of noise, this limitation seems to require a series of such networks, one for each noise level, which is impractical and even unrealistic especially in view of their prohibitive training time Burger (2013).

In this chapter, through an analysis of the interplay between the neural networks and the underlying patch distribution they seek to learn, we show how to construct a linear transform that moves natural image patches within the support of their distribution, which is then used to make a single existing deep neural network work well across all levels of Gaussian noise. In the concluding section, using the natural patch distribution invariance argument, we hint that further patch normalization may help scale down these deep neural networks by reducing their domain of definition without compromising their power.

4.2 A single neural network for all noise levels

To make a single neural network work for all noise levels, first we need to investigate the statistical regularity of the *natural patch space*, or the support of the natural patch distribution, with respect to some linear transforms: we drew 10^6 39-by-39 random patches from the *Berkeley Segmentation Dataset* (BSD500) rendered to grayscale with *Matlab*'s `rgb2gray` function. The patches were then normalized using the formula

$$\bar{p} = (p/255 - 0.5) \cdot 5 \tag{4.2}$$

provided in Burger et al. (2012) in order to conform them to these deep neural networks' training patch distribution. For each normalized patch, we computed the mean and standard deviation of its 1521 pixels and then plotted their population distribution along these two dimensions. For comparison, we did the same with the *PASCAL VOC 2012* dataset.

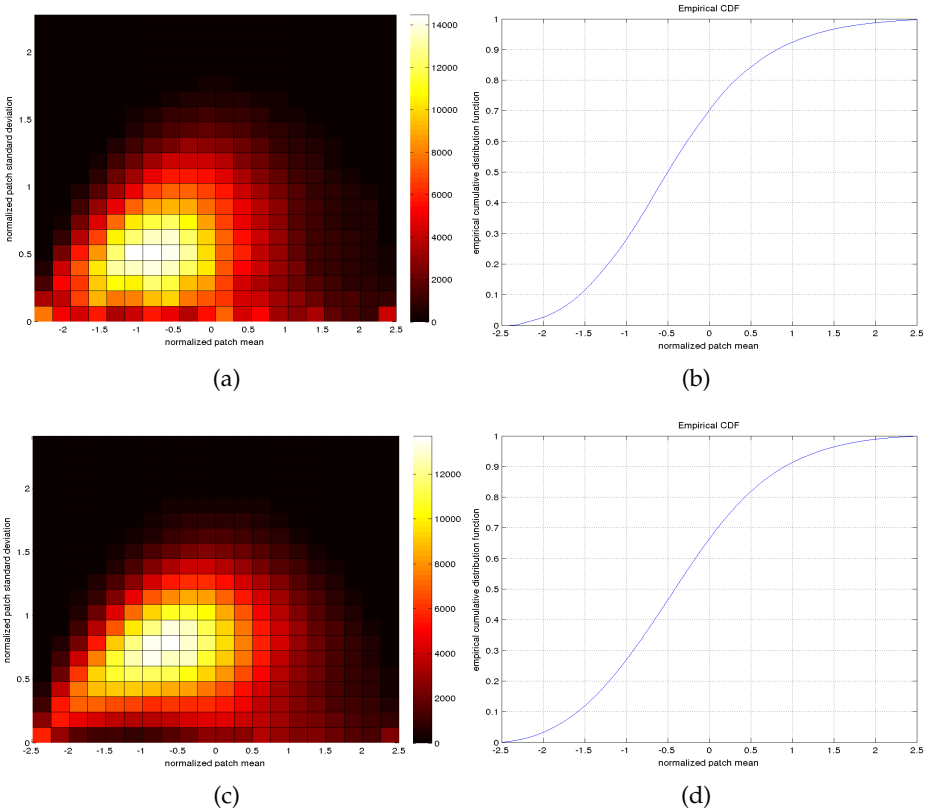


Figure 4.1 – *2D histogram of random patches from 4.1(a) the grayscale BSD500 and 4.1(c) the grayscale PASCAL VOC 2012. Their horizontal (resp. vertical) axis represents the normalized patch's mean (resp. standard variation). 4.1(d) and 4.1(b) plot their respective marginal cumulative distribution function along the patch mean. In both cases, the normalized patches have their mean concentrated around* -0.5*. Also note that at the two ends of the horizontal axis, there are two important flat patch clusters because of saturation.*

The results (Fig.4.1) show that the means of the normalized natural patches concentrate around -0.5. Even without access to the supervised pairs used in Burger et al. (2012), we can therefore expect their neural networks trained according to the criterion (4.1) to do well with patches from this neighborhood because of the sheer number of examples available. Moreover, the patch-wide variance peaking at the patch-wide means around -0.5 strongly indicates a higher tolerance for linear transforms there, that is, it is more probable for a natural patch p transformed by

$$q = a \cdot (p - s) \quad \text{with } s \text{ some constant patch and } a \geq 0$$

to remain *natural* if q's mean is close to -0.5.

To verify this conjecture that a linear transform exists which only moves patches within the support of the natural patch distribution, we designed a test that shifts the means of the patches to the same value before denoising them using a neural network, after which the shifted differences were added back individually to obtain their final, denoised versions. It is important to note that thanks to a high ratio between the patch size and σ, estimating the clean patch mean from its noisy version (Line 7 in Test) is very reliable.

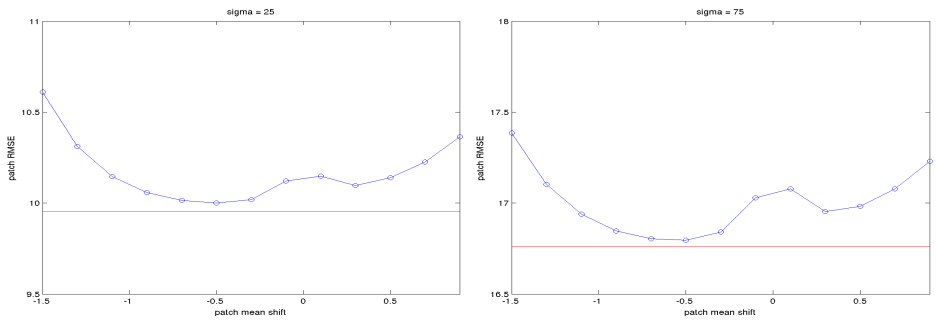
Test Patch Mean Normalization Test

1: **Input:** n clean patches p_j of dimension 39×39
2: **Output:** n denoised (resp. clean) 17×17 patches $\widehat{\mathfrak{p}}_j$ (resp. $\mathfrak{p}_j$)
3: **Parameter:** noise variance σ^2 and desired patch mean value $\mathfrak{m}$
4: **for** $j = 1$ to n **do**
5: retrieve the 17×17 clean patch $\mathfrak{p}_j$ in the center of p_j
6: generate the normalized noisy patch

$$\tilde{x}_j \leftarrow ((p_j + n_j)/255 - 0.5) \cdot 5$$

with n_j having 1521 i.i.d. Gaussian random variables $\mathcal{N}(0, \sigma^2)$ and n_j independent of $n_{j'}$ for $j \neq j'$
7: compute the patch mean shift $s_j = \frac{1}{1521} \sum_{k=1}^{1521} \tilde{x}_{jk} - \mathfrak{m}$ where $\tilde{x}_{jk}$ is the k-th pixel in $\tilde{x}_j$
8: shift the network's input $\forall k,\ \tilde{x}_{jk} \leftarrow \tilde{x}_{jk} - s_j$
9: denoise $y_j = f_\sigma(\tilde{x}_j, \theta)$ where $f_\sigma(\cdot, \theta)$ is the deep neural network Burger et al. (2012) trained with Gaussian noise standard deviation σ
10: shift the network's output by the same amount $\forall k,\ y_{jk} \leftarrow y_{jk} + s_j$ in the opposite direction
11: reverse the normalization $\widehat{\mathfrak{p}}_j \leftarrow (y_j/5 + 0.5) \cdot 255$
12: **end for**

Running the test on 10^5 39-by-39 random patches drawn from the grayscale BSD500, we computed the resultant root mean square error (RMSE). Fig.4.2 shows that regardless of the applied noise strength σ, the best empirical performance is attained with the patch mean shift set to -0.5, thereby fully confirming our supposition. In addition, observe that the optimal patch mean shift incurs surprisingly little RMSE loss, which is less than 0.1 for a noise level as high as 75.



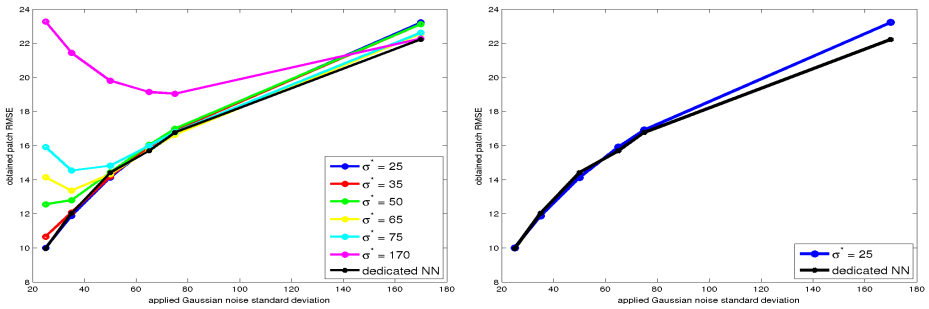
(a) (b)

Figure 4.2 – *4.2(a) (resp. 4.2(b)) plots the RMSEs resulting from the test with $\sigma = 25$ (resp. 75). The blue curve records the dedicated network's performance with the patch mean values ranging from -1.5 to 0.9. And the red line is the RMSE achieved on the same test data without patch mean shift. Other available neural networks have also been tested with very similar results.*

The previous analysis paves the way for a generic network able to handle all levels of noise. Let us use the neural network trained with the noise level σ^* for instance. To restore a noisy patch $\tilde{p}$ with noise standard deviation at σ, one multiplies $\tilde{p}$ by $\sigma^* \sigma^{-1}$, apply the patch mean shift and let the neural network operate on the normalized patch (see Algorithm). Henceforth σ^* itself becomes a parameter for further optimization, too.

Algorithm Generic Neural Network Denoising

1: **Input:** noisy patch $\tilde{p}$ of dimension 39×39 and its noise level σ
2: **Output:** denoised 17×17 patch $\widehat{\mathfrak{p}}$
3: **Parameter:** noise level σ^* of the trained neural network $f_{\sigma^*}(\cdot, \theta)$ and optimal shift $\mathfrak{m} = -0.5$
4: Scale the noise $\tilde{p} \leftarrow \sigma^* \sigma^{-1} \tilde{p}$
5: Normalize the patch $\tilde{x} \leftarrow (\tilde{p}/255 - 0.5) \cdot 5$
6: Compute the patch mean shift $s = \frac{1}{1521} \sum_{k=1}^{1521} \tilde{x}_k - \mathfrak{m}$ where $\tilde{x}_k$ is the k-th pixel in $\tilde{x}$
7: Shift the network's input mean $\forall k,\ \tilde{x}_k \leftarrow \tilde{x}_k - s$
8: Run the generic neural network denoising $y = f_{\sigma^*}(\tilde{x}, \theta)$
9: Shift the network's output by the same amount $\forall k,\ y_k \leftarrow y_k + s$ in the opposite direction
10: Reverse the normalization $\widehat{\mathfrak{p}} \leftarrow (y/5 + 0.5) \cdot 255 \cdot (\sigma^* \sigma^{-1})^{-1}$



(a) (b)

Figure 4.3 – *4.3(a) Performance discrepancy between dedicated neural networks and σ^*-indexed generic neural networks at handling different noise levels. The horizontal axis marks various test Gaussian noise levels and the vertical axis the achieved RMSEs on 10^5 random patches from BSD500. 4.3(b) singles out the best generic network with $\sigma^* = 25$.*

A comparison among various σ^*-indexed generic networks constructed this way (see Fig.4.3) shows that the one built with $\sigma^* = 25$ is the best, which is hardly surprising because a neural network can learn the most about the underlying patch space when noise is weak. Moreover, except for the extremely noisy case, one sees no tangible difference in Fig.4.3(b) between a dedicated network and the best generic network. Also observe that the higher the σ^*, the worse the resulting RMSEs at low noise levels, an expected phenomenon since a high $\sigma^* \sigma^{-1}$ exaggerates the patch-wide variation so much that resultant patches no longer remain in the natural patch space.

The implication of this analysis for future research is straightforward: training a network with the same architecture but at an even lower noise level may be rewarding. But it should also be said that too low a σ^* is not likely to work well in a strong noise environment, as already observed in Fig.4.3. Because the factor $\sigma^* \sigma^{-1}$ will then flatten all the meaningful patch-wide variations, leaving the network unable to tell one patch from another.

Put differently, what we have shown is that the trained deep network is not very different from its most informative section. Hence, one may want to train a network on mean normalized patches to improve performance,

Table 4.1 – *Comparison between dedicated neural networks and our algorithm (generic $\sigma^* = 25$)*

$\sigma = 25$	**dedicated**	**generic**
computer	**7.99**	8.10
dice	2.77	2.77
flower	**4.59**	4.63
girl	**3.38**	3.41
traffic	**9.16**	9.22
valldemossa	**11.85**	11.99
avg.	**6.62**	6.68

$\sigma = 35$	**dedicated**	**generic**
computer	**9.63**	9.82
dice	**3.29**	3.45
flower	**5.53**	5.64
girl	**3.88**	4.07
traffic	**10.81**	10.94
valldemossa	**14.21**	14.28
avg.	**7.89**	8.03

$\sigma = 50$	**dedicated**	**generic**
computer	**11.59**	11.92
dice	**4.17**	4.50
flower	**6.85**	7.06
girl	**4.60**	4.92
traffic	**12.83**	13.07
valldemossa	**16.98**	17.06
avg.	**9.50**	9.75

$\sigma = 65$	**dedicated**	**generic**
computer	**13.16**	13.82
dice	**4.83**	5.39
flower	**7.89**	8.20
girl	**5.28**	5.74
traffic	**14.32**	14.78
valldemossa	**18.67**	18.99
avg.	**10.69**	11.15

$\sigma = 75$	**dedicated**	**generic**
computer	**14.10**	14.78
dice	**5.48**	6.14
flower	**8.57**	8.96
girl	**5.66**	6.20
traffic	**15.08**	15.56
valldemossa	**19.97**	20.34
avg.	**11.47**	11.99

$\sigma = 170$	**dedicated**	**generic**
computer	**20.51**	21.81
dice	**9.23**	10.92
flower	**12.84**	13.64
girl	**8.79**	10.54
traffic	**20.71**	21.72
valldemossa	**26.42**	27.26
avg.	**16.41**	17.64

thanks to a denser data distribution. However, in so doing, one implicitly trains on a marginal distribution and thus risks losing information.

To conclude, we tested the Algorithm with $\sigma^* = 25$ on six noise-free images proposed for benchmarking various denoising algorithms (see Figure 1.5) and the results are compiled in Tab.4.1. Recall that even for $\sigma = 25$, our generic network is different from the dedicated one, in that ours involves one additional step of patch mean shift. The obtained results are thus consistent with Fig.4.2(a).

4.3 Conclusion

In this chapter, we have shown how to make a single existing neural network work well across all levels of Gaussian noise, thereby allowing

to reduce significantly the training time for a general-purpose neural network powered denoising algorithm.

To make deep neural network based algorithm more practical, the other major challenge is to reduce their sizes. This might be achieved through further patch normalization so as to reduce the neural network's input complexity. Rotation and scale invariances of natural image statistics might be used for that purpose.

A note on the size of denoising neural networks

5

Patch based denoising algorithms seek to approximate the conditional expectation of clean patches given their related noisy observations. In this chapter, we give a probabilistic account of how various algorithms approach this problem and in particular, through a conditional expectation decomposition, we argue that small neural networks can do almost as well as their large counterparts at denoising small-scale texture patterns. The analysis further indicates that self-similarity and neural networks are complementary paradigms for patch denoising, which we illustrate with an algorithm that effectively complements BM3D with small neural networks, thereby outperforming BM3D with minor additional cost.

5.1 Introduction

5.1.1 Patch denoising and its probabilistic interpretation

Patch denoising, which lies at the heart of most denoising algorithms, is concerned with estimating a noise-free patch X from a noisy observation $\tilde{Y}$

$$\tilde{Y} = Y + N$$

where Y is also noise-free and N zero-mean Gaussian noise with a known standard deviation σ. Y is usually related to X although we do not make any assumption on its nature until later. As stated in the previous chapters, the ideal filter in the sense of mean squared error (MSE) is the conditional expectation $\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}]$ because of the equality

$$\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}] = \operatorname*{argmin}_f \mathbb{E}_{\mathbb{P}}\|f(\tilde{Y}) - X\|_2^2, \quad \mathbb{P}\ a.e.$$

where the probability $\mathbb{P}$ is the triplet $(X, Y, \tilde{Y})$'s joint distribution.

When X and $\tilde{Y}$ form a Gaussian family, this conditional expectation is the classic Wiener filter Kay (1993). More generally, a version of this $\mathbb{P}$ almost surely defined function of $\tilde{Y}$ can be estimated with weakly correlated samples $(x_i, y_i)_{i\geq 1}$ of (X, Y) thanks to the equality

$$\mathbb{E}_{\mathbb{P}}[X|\tilde{Y} = \tilde{y}] = \lim_{n\to\infty} \frac{\sum_{i=1}^n p(\tilde{y}|y_i)x_i}{\sum_{i=1}^n p(\tilde{y}|y_i)}, \quad \mathbb{P}\ a.e.$$

with

$$p(\tilde{y}|y_i) \propto \exp(-\frac{\|\tilde{y} - y_i\|_2^2}{2\sigma^2}).$$

This point-wise estimator was used as an upper bound of the optimal patch denoising MSE Levin and Nadler (2011). Despite its theoretical appeal, this estimator suffers from the curse of dimensionality of a generic Monte-Carlo simulation for its inability of performing importance sampling, that is, prioritizing the samples whose y component is close to $\tilde{y}$. In addition, given the huge variety of natural patches, the mere existence of a single noise-free candidate for each noisy patch requires a data storage capacity too demanding for practical use.

An alternative is to make direct use of noisy observations. Assuming that X and Y are two disjoint sets of pixels, we may use

$$\mathbb{E}_{\mathbb{P}}[X|\tilde{Y} = \tilde{y}] = \mathbb{E}_{\mathbb{P}}[\tilde{X}|\tilde{Y} = \tilde{y}] \approx \frac{\sum_{i=1}^{n} \tilde{x}_i \mathcal{W}(\|\tilde{y} - \tilde{y}_i\|_2)}{\sum_{i=1}^{n} \mathcal{W}(\|\tilde{y} - \tilde{y}_i\|_2)}$$

with the choice of weight function $\mathcal{W}(\cdot)$ reflecting our belief in the conditional expectation's smoothness. The proof of this estimator's asymptotic consistency is generally known as Stone's theorem Devroye et al. (1996). Owing to self-similarity in natural images, this approximation scheme turns out to be very effective and was popularized by non-local means Buades et al. (2005), leading eventually to BM3D Dabov et al. (2007).

Nonetheless these nonparametric estimators are constrained by the availability of data. A potential remedy is a parametric approach. However, unlike the Wiener filter which results from a parametric data modelling, we can look for a parametric approximation of the conditional expectation, which is possible with a large and well structured class of functions parameterized by a vector-valued θ:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{P}}\|f_\theta(\tilde{Y}) - X\|_2^2 = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{\mathbb{P}}\|f_\theta(\tilde{Y}) - \mathbb{E}[X|\tilde{Y}]\|_2^2.$$

This is where multilayer feedforward neural networks come into play. Their fully-connected structure consists of a succession of non-linear hidden layers followed by a linear decoder

$$f_\theta(\cdot) = s \circ h_n \circ \cdots \circ h_1(\cdot), \quad n \geq 1$$

with

$$\forall 1 \leq l \leq n, \quad h_l(z) = \tanh(W_l z + b_l)$$

and

$$s(z) = W_{n+1} z + b_{n+1}$$

where the activation function $\tanh(\cdot)$ is applied element-wise and the parameter vector θ comprises all the connection weights and biases $(W_l, b_l)_{1 \leq l \leq n+1}$. They are tuned with stochastic gradient descent, also called backpropagation LeCun et al. (1998) Bottou (2010) due to their particular function structure. In spite of their non-convex training objective, as pointed out before, neural networks enjoy widespread use thanks to their superior practical performances Bengio (2009). Unfortunately, their are typically computationally intensive. For instance with four hidden layers having over 2000 units each, denoising neural networks Burger et al.

(2012) effectively require tens of millions of multiplication operations per pixel when denoising a grayscale image.

Since it is recognized Burger et al. (2012) that as denoising algorithms, neural networks do not always dominate BM3D, it was proposed Burger et al. (2013) to train one more neural network of comparable size which takes in the denoised patches from both BM3D and neural networks in addition to the original noisy patch and outputs a combined result. This method improves both algorithms because, again in view of the conditional expectation, we have for any random triplet $(X, \tilde{Y}, \tilde{H})$

$$\mathbb{E}_{\mathbb{P}}\|X - \mathbb{E}_{\mathbb{P}}[X|\tilde{Y}]\|_2^2 - \mathbb{E}_{\mathbb{P}}\|X - \mathbb{E}_{\mathbb{P}}[X|\tilde{Y}, \tilde{H}]\|_2^2 = \mathbb{E}_{\mathbb{P}}\|\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}, \tilde{H}] - \mathbb{E}_{\mathbb{P}}[X|\tilde{Y}]\|_2^2 \geq 0,$$

that is, more information helps lower the theoretical MSE bound. However, the proposed approach doubles the already heavy computational load.

5.1.2 Our contribution

In this chapter, we investigate whether it is possible to scale down the neural networks while preserving their performance. After observing that they constitute a better alternative than self-similarity for small-scale texture pattern denoising, we argue that large neural networks and their heavy computational cost may be unnecessary for this specific task through a conditional expectation decomposition. This analysis further indicates that self-similarity and neural networks are complementary approaches to patch denoising, which we illustrate with a light-weight algorithm complementing BM3D with small neural networks geared towards granular texture denoising, which achieves better performance than either individually at minor additional cost.

5.2 Small-scale texture pattern denoising

5.2.1 A conditional expectation decomposition

Patterns in natural images tend to repeat themselves. Non-local means Buades et al. (2005) and BM3D Dabov et al. (2007) use this prior to find similar neighboring patches to denoise. Further efforts have been made to characterize the similarity in terms of orientation (Chapter 1, 6) or even finer features Zoran and Weiss (2011). However, no convincing mechanism exists in these algorithms to handle non-repetitive small-scale texture patterns, which can be a problem in denoising a picture taken against a background of lush foliage for example.

Neural networks thus offer a valuable solution because in texture areas, the interactions among pixels are of short range by nature. Formally, let $\tilde{Z}, \tilde{Y}, \tilde{X}$ be three nested noisy patches with strictly decreasing domains. With $\tilde{X}$'s noise-free state denoted as X, it seems reasonable to posit that X and $\tilde{Z} \setminus \tilde{Y}$ are independent conditional on $\tilde{Y}$ with the boundary $\tilde{Y} \setminus \tilde{X}$ wide enough. It means that under such a generative probability, we have

$$\mathbb{E}[X|\tilde{Z}] = \mathbb{E}[X|\tilde{Y}].$$

But the same cannot be said of all noisy patterns. Consider for instance flat patches with identical grayscale intensity. Then, with d_x, d_y and d_o

denoted as the number of pixels in X, $\tilde{Y}$ and $\tilde{O} := \tilde{Z} \setminus \tilde{Y}$, the joint law of $X, \tilde{Y}$ and $\tilde{O}$ is proportional to

$$\exp(-\frac{\|\sqrt{d_x^{-1}}\|x\|_2 1_{d_o} - \tilde{o}\|_2^2}{2\sigma^2}) \exp(-\frac{\|\sqrt{d_x^{-1}}\|x\|_2 1_{d_y} - \tilde{y}\|_2^2}{2\sigma^2}) \mu(dx) d\tilde{y} d\tilde{o}$$

where 1_{d_o} represents the d_o-dimensional vector with identical entries equal to 1 and $\mu(\cdot)$ a probability supported on the d_x-dimensional segment $[0 \cdot 1_{dx}, 255 \cdot 1_{d_x}]$. Thus, the conditional independence of X and $\tilde{O}$ given $\tilde{Y}$ no longer holds.

Mathematically speaking, neural networks are trained on a more complex distribution $\mathbb{P}$. If $\mathbb{P}$ can be decomposed as $\alpha\mathbb{M} + (1-\alpha)\mathbb{Q}$ for some $\alpha \in (0,1)$ with $\mathbb{M}$ having short range properties and $\mathbb{Q}$ not, a connection of the two conditional expectations can be made explicitly.

Theorem 5.1 *Let $\mathbb{P}, \mathbb{M}, \mathbb{Q}$ be three probabilities defined on a common measurable space satisfying $\mathbb{P} = \alpha\mathbb{M} + (1-\alpha)\mathbb{Q}$ for some $\alpha \in (0,1)$. Let U, V be two random vectors with $\mathbb{E}_{\mathbb{P}}\|U\|_1 < +\infty$ defined on the same space. Then we have*

$$\mathbb{E}_{\mathbb{P}}[U|V] = \alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{M}}[U|V] + (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{Q}}[U|V], \quad \mathbb{P}\ a.e. \tag{5.1}$$

where $\frac{d\mathbb{M}}{d\mathbb{P}}$ and $\frac{d\mathbb{Q}}{d\mathbb{P}}$ are the Radon-Nikodym derivatives of $\mathbb{M}$ and $\mathbb{Q}$ with respect to $\mathbb{P}$.

Proof: by the definition of conditional expectation, we have for any bounded Borel function $\phi(\cdot)$

$$\begin{aligned}
&\mathbb{E}_{\mathbb{P}}[\mathbb{E}_{\mathbb{P}}[U|V]\phi(V)] && (5.2)\\
=&\mathbb{E}_{\mathbb{P}}[U\phi(V)]\\
=&\alpha\mathbb{E}_{\mathbb{M}}[U\phi(V)] + (1-\alpha)\mathbb{E}_{\mathbb{Q}}[U\phi(V)]\\
=&\alpha\mathbb{E}_{\mathbb{M}}[\mathbb{E}_{\mathbb{M}}[U|V]\phi(V)] + (1-\alpha)\mathbb{E}_{\mathbb{Q}}[\mathbb{E}_{\mathbb{Q}}[U|V]\phi(V)]\\
=&\alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}\mathbb{E}_{\mathbb{M}}[U|V]\phi(V)] + (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}\mathbb{E}_{\mathbb{Q}}[U|V]\phi(V)] && (5.3)\\
=&\mathbb{E}_{\mathbb{P}}[\Big(\alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{M}}[U|V] + (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{Q}}[U|V]\Big)\phi(V)].
\end{aligned}$$

The equation (5.3) holds because $\mathbb{M}$ and $\mathbb{Q}$ are absolutely continuous with respect to $\mathbb{P}$ by construction. As a result, if we define $D(V)$ to be the first coordinate of the random vector

$$\alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{M}}[U|V] + (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}|V]\mathbb{E}_{\mathbb{Q}}[U|V] - \mathbb{E}_{\mathbb{P}}[U|V], \tag{5.4}$$

then for all $n \in \mathbb{N}$

$$\mathbb{E}_{\mathbb{P}}[D(V)1_{D(V)<-n^{-1}}] = \mathbb{E}_{\mathbb{P}}[D(V)1_{D(V)>n^{-1}}] = 0$$

because both $1_{D(\cdot)>n^{-1}}$ and $1_{D(\cdot)<-n^{-1}}$ are bounded. So we deduce

$$n^{-1}\mathbb{P}(|D(V)| > n^{-1}) \leq \mathbb{E}_{\mathbb{P}}[|D(V)|1_{|D(V)|>n^{-1}}] = 0$$

because

$$\mathbb{E}_{\mathbb{P}}[|D(V)|1_{|D(V)|>n^{-1}}] = \mathbb{E}_{\mathbb{P}}[D(V)1_{D(V)>n^{-1}}] - \mathbb{E}_{\mathbb{P}}[D(V)1_{D(V)<-n^{-1}}],$$

which implies

$$\mathbb{P}(|D(V)| > 0) = \lim_{n\to\infty} \mathbb{P}(|D(V)| > n^{-1}) = 0 \Leftrightarrow \mathbb{P}(D(V) = 0) = 1.$$

The same reasoning applies to the other coordinates of the random vector (5.4). Hence the probability for this random vector to vanish is equal to 1. □

Since the classic likelihood ratios Devroye et al. (1996) found in the conditional expectation decomposition are positive and satisfy

$$\alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}|V] + (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}|V] = \mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{P}}{d\mathbb{P}}|V] = 1,$$

we will denote them as

$$\gamma(V) := \alpha\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{M}}{d\mathbb{P}}|V],\ 1-\gamma(V) := (1-\alpha)\mathbb{E}_{\mathbb{P}}[\frac{d\mathbb{Q}}{d\mathbb{P}}|V]$$

for notational convenience. In our context, the theorem implies

$$\mathbb{E}_{\mathbb{P}}[X|\tilde{X}] = \gamma(\tilde{X})\mathbb{E}_{\mathbb{M}}[X|\tilde{X}] + (1-\gamma(\tilde{X}))\mathbb{E}_{\mathbb{Q}}[X|\tilde{X}]$$
$$\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}] = \gamma(\tilde{Y})\mathbb{E}_{\mathbb{M}}[X|\tilde{X}] + (1-\gamma(\tilde{Y}))\mathbb{E}_{\mathbb{Q}}[X|\tilde{Y}].$$

For an observation whose likelihood ratios strongly favor the short range hypothesis for both $\tilde{X}$ and $\tilde{Y}$, that is,

$$\gamma(\tilde{X}) \approx \gamma(\tilde{Y}) \approx 1$$

$\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}]$ can thus be approximated by $\mathbb{E}_{\mathbb{P}}[X|\tilde{X}]$, thereby enabling an input size reduction.

Note however that short range interaction is only an idealized depiction of texture formation process. The objects composing a random texture are usually of regular shape when viewed at a close distance relative to their dimensions. Only when viewed from afar can they be adequately accounted for by such a modelling assumption Portilla and Simoncelli (2000). This observation motivates us to investigate the possibility of scaling down the neural networks for denoising small-scale texture patterns.

5.2.2 Small neural networks for small-scale texture denoising

We set up three identical neural networks which act on a 7-by-7 noisy input to estimate its central 3-by-3 patch. Comprising three hidden layers with 147 units each, their architecture can be summarized as 49-147-147-147-9. They were trained following the same procedure as explained in Burger et al. (2012) under Gaussian noise standard deviation set to 10, 25, 35 respectively in order to match the published ones, whose architectures are 441-2047-2047-2047-2047-81, 1521-3072-3072-2559-2047-289 and 1521-3071-3071-2559-2047-289.

Two images were selected for testing purposes. The first 746-by-264 image was cropped from a standard *Kodak* PhotoCD benchmark image

Figure 5.1 – *The two test images*

rendered to grayscale with *matlab*'s `rgb2gray` function. It was chosen because of its rich small-scale texture content. The second 1280-by-1280 image contains artificially generated structured patterns. Neither image is in the Pascal VOC dataset from which our training and validation images were drawn.

Table 5.1 – *RMSE comparison between BM3D, small and large neural networks*

$\sigma = 10$	**BM3D**	**small**	**large**
structure	**1.31**	2.42	1.65
texture	8.11	7.98	**7.86**

$\sigma = 25$	**BM3D**	**small**	**large**
structure	2.83	4.48	**2.48**
texture	14.91	14.48	**14.16**

$\sigma = 35$	**BM3D**	**small**	**large**
structure	3.74	5.84	**3.06**
texture	17.71	17.30	**16.86**

Table 5.1 indicates that small neural networks do almost as well as their large counterparts at denoising small-scale texture patterns. But their smallness does limit what they can do for highly structured patterns. On the other hand, BM3D underperformed both neural networks where self-similarity was lacking. BM3D's 39-by-39 search window, vis-à-vis the large neural networks' input patch size (21-by-21 for $\sigma = 10$ and 39-by-39 for the rest) also shows that it is more effective when noise is lower. Moreover, computationally speaking, to denoise a grayscale image, these small neural networks require roughly 5×10^4 operations per pixel, which is more than 500 times cheaper than their large counterparts and makes them on a par with BM3D because BM3D needs 4×10^4 operations per pixel if all of its transforms are implemented with a time complexity equal to $N \log_2 N$ Dabov et al. (2007).

To gain a further understanding of the texture patterns that satisfy our hypothesis, we downloaded several texture images and downsampled them to simulate a zoom out effect. Unsurprisingly, patterns with highly varying intensity levels turned out best fit our modelling assumption, because they look more like a noise process locally (see Fig.5.2).

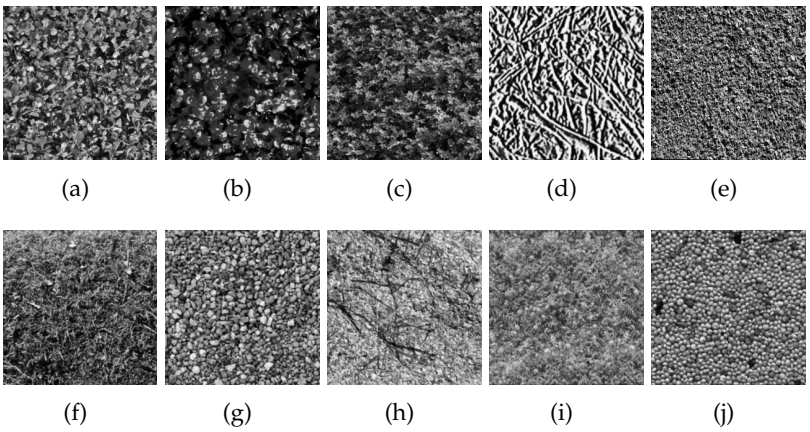


Figure 5.2 – *(a)(b)(c)(d)(e) are the texture patterns for which similar results as in Table 5.1 were observed. The same cannot be said for (f)(g)(h)(i)(j).*

5.3 Complementing BM3D with small neural networks

In this section, we present an algorithm named SSaNN, for self-similarity and neural network, which relies on a Gaussian mixture classification device to switch between self-similarity based BM3D and neural networks.

Let $\tilde{X}, \tilde{Y}, \tilde{Z}$ be three nested noisy patches with strictly growing domains. Let X be the noise-free state of $\tilde{X}$. The previous section shows

$$\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}] \approx \mathbb{E}_{\mathbb{M}}[X|\tilde{Z}] \approx \mathbb{E}_{\mathbb{P}}[X|\tilde{Z}]$$

if $\tilde{Z}$ is generated under $\mathbb{M}$. We can then combine the strength of BM3D and small neural networks in a principled framework. The key is to observe that the two algorithms underperform in different but complementary scenarios. In this circumstance, the conditional expectation decomposition is ideally suited to guiding us because the formula

$$\mathbb{E}_{\mathbb{P}}[X|\tilde{Z}] = \gamma(\tilde{Z})\mathbb{E}_{\mathbb{M}}[X|\tilde{Z}] + (1 - \gamma(\tilde{Z}))\mathbb{E}_{\mathbb{Q}}[X|\tilde{Z}]$$

suggests replacing the two built-in filters $\mathbb{E}_{\mathbb{M}}[X|\tilde{Z}]$ and $\mathbb{E}_{\mathbb{Q}}[X|\tilde{Z}]$ of large neural networks by their less costly alternatives $\mathbb{E}_{\mathbb{P}}[X|\tilde{Y}]$ and BM3D respectively. Furthermore, the same formula shows that conditional expectation mechanically involves a pair of related likelihood ratios, which can also be estimated with the help of an external device for probabilistic patch space modelling.

To this end, we may use the 20-component Gaussian mixture proposed in Chapter 1. The restoration of regularly structured patterns, embodied here by the flat and oriented classes, can benefit from more extensive observation. It is the basic premise of self-similarity based algorithms, which is also supported by our previous analysis and comparison between BM3D and the neural networks. Therefore, within the mixture, the law $\mathbb{M}$ is best interpreted as the conditional probability attached to the texture class.

Given a noisy image and its noisy level, our algorithm first produces two denoised versions by the small neural network trained under the same noise condition and BM3D. Then it extends the noisy image in order to

form a one-to-one mapping between the 7-by-7 patches in the extended image and the pixels in the original noisy image by associating a patch with its central pixel. Next the EM algorithm is run three times with the initialized Gaussian mixture on the extended noisy image to obtain for each 7-by-7 patch its posterior probability of belonging to the texture class, which is then attached to its associated pixel. Finally, the algorithm combines the two denoised image versions using the posterior probabilities as their respective weight. For a faster execution, one could apply the Bayes rule Devroye et al. (1996) (Chapter 1) to first classify the pixels into either texture or non-texture, and then choose to denoise them with the neural network or BM3D. Since this latter version incurs relatively little RMSE loss (less than 0.5 across different images and noise levels), we only report its results in Table 5.2. Also note that instead of 18 oriented models originally designed for the mixture 1, we put in 6 which costs an additional 2.8×10^4 operations per pixel as a result of the three EM iterations. The parameters t_{flat}=10^4 and t_{orient}=1.2 were fixed by testing the algorithm on images separate from the ones reported in the Table.

(SSaNN) Combining BM3D and small neural networks

1: **Input:** noisy image $\tilde{I}$, a Gaussian mixture initialized on 7-by-7 patches
2: **Output:** denoised image
3: **Parameter:** Gaussian noise standard deviation σ
4: Denoise $\tilde{I}$ with BM3D and the neural network to get I_B and I_N.
5: Extend $\tilde{I}$ to $\bar{I}$ so that $\bar{I}$'s 7-by-7 patches form one-to-one mapping with $\tilde{I}$'s pixels.
6: Run the expectation-maximization algorithm on $\bar{I}$'s patches to obtain the posterior probability of each patch of belonging to the texture class. Assign them to their central pixels so as to form T_S.
7: Soft combination results from $I_B \circ (1 - T_S) + I_N \circ T_S$ where $\circ$ denotes the element-wise product.
8: Form another matrix T_H of the same size as T_S such that its entry at (x, y) is $T_H(x, y) = 1_{T_S(x,y)>0.5}$.
9: Hard combination results from $I_B \circ (1 - T_H) + I_N \circ T_H$.

Table 5.2 demonstrates that our algorithm effectively combines the respective strength of BM3D and small neural networks, thereby improving BM3D on images with small-scale texture content. However, since the performance of both the substituted filters and the classification device is negatively affected by noise (see Fig.5.3 and Chapter 1), the relative performance gap between SSaNN and the large neural networks increases with noise too.

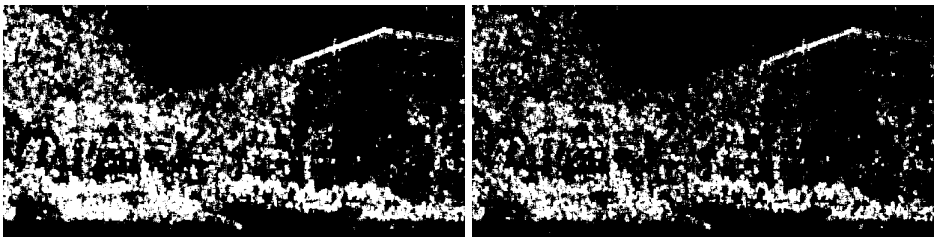
Table 5.2 – *RMSE comparison between BM3D, small and large neural networks and our algorithm*

$\sigma = 10$	**small**	**BM3D**	**SSaNN**	**large**
computer	4.97	4.66	**4.63**	4.77
dice	2.58	**1.79**	**1.79**	1.89
flower	3.19	2.83	2.81	**2.77**
girl	2.93	**2.32**	**2.32**	2.41
traffic	5.67	5.64	5.54	**5.52**
valldemossa	6.55	6.61	6.51	**6.48**

$\sigma = 25$	**small**	**BM3D**	**SSaNN**	**large**
computer	8.88	8.15	**8.13**	8.20
dice	4.81	3.04	3.04	**2.85**
flower	5.61	5.15	5.12	**4.85**
girl	5.08	3.64	3.64	**3.46**
traffic	10.16	9.94	9.79	**9.55**
valldemossa	12.22	12.28	12.06	**11.85**

$\sigma = 35$	**small**	**BM3D**	**SSaNN**	**large**
computer	10.86	9.88	**9.87**	9.88
dice	6.24	3.80	3.80	**3.39**
flower	6.99	6.40	6.38	**6.00**
girl	6.39	4.41	4.41	**4.01**
traffic	12.26	11.85	11.75	**11.36**
valldemossa	14.81	14.74	14.53	**14.24**

(a) (b)



(c) (d)

Figure 5.3 – *Highlighted in white are the pixels in (a) classified as texture at noise standard deviation (b) $\sigma = 10$ (c) $\sigma = 25$ and (d) $\sigma = 35$.*

Part III. Reproducible research

The third part of this thesis consists of reproducible research. In addition to the algorithm introduced in Chapter 2, the Viola-Jones face detection algorithm is studied in detail and implemented, which includes a derivation of the core Adaboost feature selection algorithm.

6 E-PLE : an algorithm for image inpainting

In this chapter, we present a probabilistic view of an existing image inpainting algorithm and propose several theoretical and numerical improvements based on an effective use of Gaussian mixture.

6.1 Introduction

Inpainting is an interpolation technique developed for repairing a partially masked image with information present in the visible parts of the same image.

Historically, one of the first works in the field Masnou and Morel (1998) proposed to connect level lines by minimizing a curvature functional due to their link made clear by the co-area formula. Later a total variation approach Shen and Chan (2002) was introduced along a similar line whose success comes from its insightful choice of functional space which avoids the blur that could be created by a more regular space such as H^1 under an otherwise identical optimization schema. The subject has since gained some popularity and inspires Bertalmio et al. (2000) where a high order PDE is used to propagate structural information to fill in relatively small gaps. To infer missing textural content, a similarity driven algorithm Efros and Leung (1999) is devised. The same idea has spawned an effective image processing paradigm Buades et al. (2005), Dabov et al. (2007), Lebrun et al. (2013b). Building on these developments on structure and texture inpainting, some efforts Elad et al. (2005), Bertalmio et al. (2003) have been made to unite these two by performing one preliminary step to separate two types of content before carrying out their respective dedicated procedure.

Another direction of research initiated in Aharon et al. (2005), Elad and Aharon (2006) targets an overcomplete dictionary for sparse representation of image patches. The orientation based K-LLD for image denoising Chatterjee and Milanfar (2009) is another example. In Yu et al. (2012) a similar algorithm, called PLE, was proposed but intended to solve generic image related inverse problems. In a recent development Zoran and Weiss (2011), a Gaussian mixture patch prior modelling is put forth with a new optimisation schema, which produces impressive results.

In this chapter, motivated in part by the works of Chatterjee and Milanfar (2009), Yu et al. (2012), Zoran and Weiss (2011), we present E-PLE, or Enhanced PLE. Using a specialized Gaussian mixture initialized with real-world images, we adapt expectation maximization (EM) algorithm Dempster et al. (1977) to this particular setting and show its improved performance at inpainting.

Section 2 summarizes PLE. An account of E-PLE is provided in section 3. Section 4 presents the new algorithm outline, followed by several comparative empirical studies in section 5. The appendix is devoted to the EM algorithm.

6.2 PLE

In this section, we describe PLE Yu et al. (2012) to highlight its difference with E-PLE. PLE begins with a number of full-rank Gaussian directional models built with synthetic samples. Then one additional model is added using DCT as its basis to account for textural patches. All these models share the same mean vector and covariance eigenvalues, which are arbitrarily fixed (see algorithm 8).

Algorithm 8 PLE initialization

Parameter: number of Gaussian models K, patch dimension $\kappa \times \kappa$.
for $k = 0$ to $K - 2$ **do**

Create and sample synthetic images

1. Create a binary image B of size 100×100 valued in $\{0, 255\}$ with two sets $\{(r, u) : \ B(r, u) = 0\}$ and $\{(r, u) : \ B(r, u) = 255\}$ separated by a straight line inclined at $\frac{k}{K-1}\pi$ passing through the center of the image.
2. Blur B with Gaussian kernels of different standard deviations $(\sigma_b)_{1 \leq b \leq 4}$: $\sigma_b = 2b$ for all b.
3. Draw a large number of $\kappa \times \kappa$ patches from these blurred images to form the patch set $\mathcal{P}_k$.

Compute the statistics

1. Estimate the model mean and covariance:

$$\boldsymbol{\mu}_k = \frac{1}{|\mathcal{P}_k|} \sum_{P \in \mathcal{P}_k} P, \quad \boldsymbol{\Sigma}_k = \frac{1}{|\mathcal{P}_k|} \sum_{P \in \mathcal{P}_k} (P - \boldsymbol{\mu}_k)(P - \boldsymbol{\mu}_k)^T.$$

2. Set $\boldsymbol{\mu}_k = 0$.
3. Define the k-th directional basis V_k using the spectral decomposition $\boldsymbol{\Sigma}_k = V_k \Lambda_k V_k^T$. Replace the first leading eigenvector in V_k by a normalized constant component and apply Gram-Schmitt to orthogonalize the remaining vectors. [1]

end for
Set up a textural model using DCT as its basis. Set its model mean to zero.
Take a sequence of κ^2 positive numbers of exponential decay (a working example: $m \in [0, \kappa^2 - 1] \cap \mathbb{Z} \mapsto 2^{20.5 - 0.5m}$) and make them the eigenvalues of all K Gaussian models just built.

1. The implemented PLE leaves out both component substitution and basis orthogonalization because they can cause numerical instability as it is difficult to tell whether a set

Assume that there are K models in all. For each patch to restore, PLE produces K estimates under individual model assumption and keeps the one with the highest conditional probability to have both the observation and its estimate. This patch is assigned to the same model.

Finally, all the models are updated with their assigned estimates. The last two steps, called estimation and maximization by the paper, are then repeated several times before the algorithm terminates (see algorithm 9).

Algorithm 9 PLE

Input: a masked gray image $\tilde{U}$, its mask M.
Parameter: number of PLE iterations S.
Run algorithm 8. Extract all $\kappa \times \kappa$ patches from $\tilde{U}$ and their associated masks from M, the collection of which is denoted by $\tilde{\mathcal{P}}$ and $\mathcal{M}$. With $|\tilde{\mathcal{P}}| = |\mathcal{M}|$, the i-th observed patch and its mask are $\tilde{P}_i$ and $\mathfrak{M}_i$.
for $t = 1$ to S **do**
Estimation:

1. Filter the patch under K model assumptions:

$$\begin{aligned}\forall (i,k),\quad \widehat{P}_i^{(k)} &= \operatorname*{argmax}_P p(P|\tilde{P}_i, \boldsymbol{\mu}_{k,t-1}, \boldsymbol{\Sigma}_{k,t-1}) \\ &= \operatorname*{argmax}_P p(P, \tilde{P}_i|\boldsymbol{\mu}_{k,t-1}, \boldsymbol{\Sigma}_{k,t-1}) \\ &= \operatorname*{argmin}_P \Big(\frac{\|\mathfrak{M}_i P - \tilde{P}_i\|^2}{\sigma^2} + (P-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_{k,t-1}^{-1} (P-\boldsymbol{\mu}_k) \Big).\end{aligned}$$

2. Select a model for each patch:

$$\begin{aligned}k_i &= \operatorname*{argmax}_{0\le k\le K-1} p(\widehat{P}_i^{(k)}, \tilde{P}_i|\boldsymbol{\mu}_{k,t-1}, \boldsymbol{\Sigma}_{k,t-1}) \qquad (6.1)\\ &= \operatorname*{argmin}_{0\le k\le K-1} \Big(\frac{\|\mathfrak{M}_i \widehat{P}_i^{(k)} - \tilde{P}_i\|^2}{\sigma^2} + (\widehat{P}_i^{(k)} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_{k,t-1}^{-1} (\widehat{P}_i^{(k)} - \boldsymbol{\mu}_k) + \ln\det \boldsymbol{\Sigma}_{k,t-1} \Big)\end{aligned}$$

which leads to its estimate $\widehat{P}_i = \widehat{P}_i^{(k_i)}$ and assignment to the k_i-th model. [2]

Maximization: Denote $\mathcal{Q}_k$ the set of estimated patches attributed to the k-th model.
for $k = 0$ to $K-1$ **do**
Estimate the model mean and covariance:

$$\boldsymbol{\mu}_{k,t} = \frac{1}{|\mathcal{Q}_k|} \sum_{P\in\mathcal{Q}_k} P, \quad \boldsymbol{\Sigma}_{k,t} = \frac{1}{|\mathcal{Q}_k|} \sum_{P\in\mathcal{Q}_k} (P-\boldsymbol{\mu}_{k,t})(P-\boldsymbol{\mu}_{k,t})^T + \epsilon I$$

where ϵ is a small positive number to ensure the definiteness of $\boldsymbol{\Sigma}_{k,t}$.
end for
end for
Assign equal weights to all restored patches and recover the image.

of vectors are collinear with the computer's limited precision. With constant components removed from the directional bases, PLE could discriminate better.

2. It would be more natural to incorporate at this stage what we know from the observations. Experiments confirmed that the algorithm yielded better results if the estimated pixels were replaced with the visible ones wherever possible. Hence, we implemented PLE with this additional step.

However, the Gaussian model used by PLE lacks the mixing weights $w_.$ for it to be a mixture

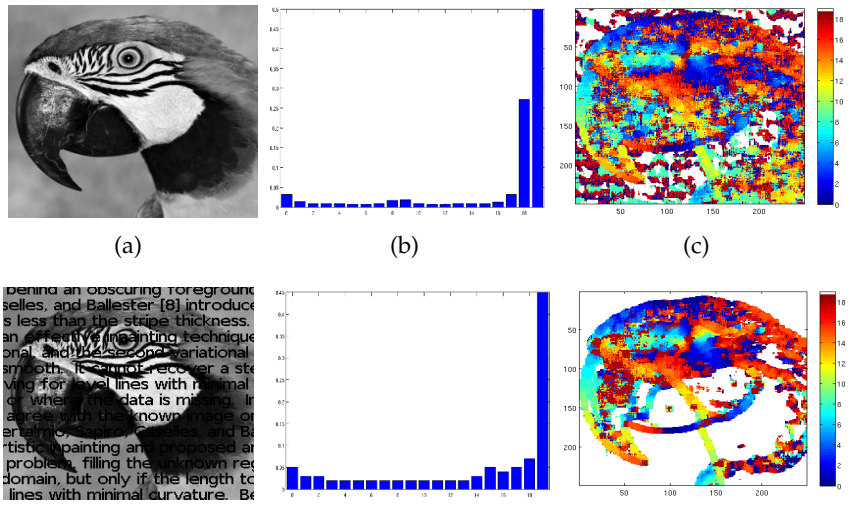
$$p(P) = \sum_{k=1}^{N} w_k \mathcal{N}(P \mid \mu_k, \Sigma_k), \tag{6.2}$$

of which the synthetic image sampling cannot produce an estimate. Thus algorithm 9 is not an EM, a class of algorithms known to increase the likelihood of a mixture over iterations Dempster et al. (1977). The absence of the mixing weights to knit the models also implies that the patch assignment step (6.1) is not statistically founded. Moreover, the use of full rank covariance matrix to model directional patterns is questionable as the synthetic samples clearly lie in a lower dimensional space (see 1.1)

6.3 E-PLE

6.3.1 Masked patch classification and adaptive filtering

To set up the Gaussian mixture for E-PLE, we follow 1.1 and feed it with real-world data (see algorithm 1) so as to shorten the algorithm's learning phase, which is carried out by a version of EM developed for our partially observed data (see appendix). A patch is then classified using the Bayes rule (1.6).



(a) (b) (c)

(d) (e) (f)

Figure 6.1 – *Masked patch classification with EM. (a) original image (d) masked image (b) initial mixing weights (e) mixing weights after three EM iterations (c) patch map formed after one EM iteration (f) patch map formed after three EM iterations. The flat patches are painted white.*

If a patch $\tilde{P}$ is classified to the k-th model

$$\tilde{P} = F_k c + \mu_k + \sigma N$$

where F_k, c, μ_k, σ and N denote its factor loading matrix, random coefficient, model mean, noise standard deviation and a standard Gaussian random vector independent of c, Tikhonov regularization can be applied

to construct an estimator. Assume without loss of generality that $\boldsymbol{F}_k$'s columns $(\boldsymbol{F}_k^{(m)})_{1\leq m\leq l_k}$ are the orthogonal leading eigenvectors of the covariance matrix $\boldsymbol{F}_k\boldsymbol{F}_k^T$. Then the following Wiener filtering schema with an adjustable parameter ξ controlling the degree of data fit

$$\begin{aligned}\widehat{P} &= \underset{\exists\boldsymbol{\beta}, P=\boldsymbol{F}_k\boldsymbol{\beta}+\boldsymbol{\mu}_k}{\operatorname{argmin}} \sum_{m=1}^{l_k} \|\boldsymbol{F}_k^{(m)}\|^{-2}\big\langle P-\boldsymbol{\mu}_k, \|\boldsymbol{F}_k^{(m)}\|^{-1}\boldsymbol{F}_k^{(m)}\big\rangle^2 + \xi\|\mathfrak{M}P-\tilde{P}\|^2 \\ &= \underset{\exists\boldsymbol{\beta}, P=\boldsymbol{F}_k\boldsymbol{\beta}+\boldsymbol{\mu}_k}{\operatorname{argmin}} \sum_{m=1}^{l_k} \big\langle P-\boldsymbol{\mu}_k, \|\boldsymbol{F}_k^{(m)}\|^{-2}\boldsymbol{F}_k^{(m)}\big\rangle^2 + \xi\|\mathfrak{M}(P-\boldsymbol{\mu}_k-\tilde{P}+\boldsymbol{\mu}_k)\|^2\end{aligned}$$

defines a ξ-indexed mapping $\tilde{P}\in\mathbb{R}^{\kappa^2}\mapsto\widehat{P}\in\mathbb{R}^{\kappa^2}$. A much neater formulation of the same problem can be obtained with some additional auxiliary variables

$$\begin{aligned}\boldsymbol{\beta} &= [\beta_1,\cdots,\beta_{l_k}]^T, \quad \beta_m = \big\langle P-\boldsymbol{\mu}_k, \|\boldsymbol{F}_k^{(m)}\|^{-2}\boldsymbol{F}_k^{(m)}\big\rangle \\ \tilde{\boldsymbol{\beta}} &= [\tilde{\beta}_1,\cdots,\tilde{\beta}_{l_k}]^T, \quad \tilde{\beta}_m = \big\langle \tilde{P}-\boldsymbol{\mu}_k, \|\boldsymbol{F}_k^{(m)}\|^{-2}\boldsymbol{F}_k^{(m)}\big\rangle.\end{aligned}$$

Now it follows:

$$\widehat{P} = \boldsymbol{F}_k\boldsymbol{\beta}_* + \boldsymbol{\mu}_k, \quad \boldsymbol{\beta}_* = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\boldsymbol{\beta}\|^2 + \xi\|\mathfrak{M}\boldsymbol{F}_k(\boldsymbol{\beta}-\tilde{\boldsymbol{\beta}}) - \mathfrak{M}R_{\tilde{P}}\|^2$$

with the residual:

$$R_{\tilde{P}} = \tilde{P} - \boldsymbol{\mu}_k - \boldsymbol{F}_k\tilde{\boldsymbol{\beta}}.$$

The solution to this quadratic minimization problem is straightforward

$$\boldsymbol{\beta}_* = \xi(I_{l_k} + \xi\boldsymbol{F}_k^T\mathfrak{M}\boldsymbol{F}_k)^{-1}\boldsymbol{F}_k^T\mathfrak{M}(\tilde{P}-\boldsymbol{\mu}_k).$$

Hence the linear estimator

$$\widehat{P} = \xi\boldsymbol{F}_k(I_{l_k} + \xi\boldsymbol{F}_k^T\mathfrak{M}\boldsymbol{F}_k)^{-1}\boldsymbol{F}_k^T\mathfrak{M}(\tilde{P}-\boldsymbol{\mu}_k) + \boldsymbol{\mu}_k. \tag{6.3}$$

Thanks to the identity I_{l_k}, the matrix inversion is well defined. In addition, because of the linear filter's symmetric form, the factor orthogonalization in $\boldsymbol{F}_k$, otherwise required to meet the assumption of the analysis, can be effectively avoided.

6.3.2 Algorithm outline

A recap of E-PLE. First, a Gaussian factor mixture is set up using natural images. Next, EM is called upon to infer its parameters from the image to inpaint. Finally, adaptive linear filters ((6.3)) are used to restore patches and hence the image.

For a color image, three color channels can be restored separately before forming the final result. One way to speed up the algorithm in this case however is to make EM only run on one channel and use the resulting patch map to guide the other two. It is the adopted approach in the current implementation Wang (2013a).

Algorithm 10 E-PLE

Input: a masked gray image $\tilde{U}$, its mask M.
Parameter: number of EM iterations S.
Run algorithm 1. Extract all 8×8 patches from $\tilde{U}$ and their masks from M, the collection of which are denoted by $\tilde{\mathcal{P}}$ and $\mathcal{M}$. With $|\tilde{\mathcal{P}}| = |\mathcal{M}| = N$, observation i and its mask are denoted by $\tilde{P}_i$ and $\mathfrak{M}_i$.
for $t = 1$ to S **do**
Expectation:

1. Compute the mean and covariance of the coefficient posteriors: $\forall 1 \leq i \leq N,\ 0 \leq s_i \leq K-1$,

$$\boldsymbol{\Sigma}_{c_i|s_i} = \Big(\frac{\boldsymbol{F}_{s_i,t-1}^T \mathfrak{M}_i \boldsymbol{F}_{s_i,t-1}}{\sigma_{t-1}^2} + I\Big)^{-1},\ \boldsymbol{\mu}_{c_i|s_i} = \boldsymbol{\Sigma}_{c_i|s_i} \frac{\boldsymbol{F}_{s_i,t-1}^T (\tilde{P}_i - \mathfrak{M}_i \boldsymbol{\mu}_{s_i,t-1})}{\sigma_{t-1}^2}.$$

The parameter set Θ_{t-1} comprises $(\boldsymbol{w}_{k,t-1}, \boldsymbol{F}_{k,t-1}, \boldsymbol{\mu}_{k,t-1})_{0 \leq k \leq K-1}$ and σ_{t-1} for $1 \leq t \leq S$. The couple $(\boldsymbol{\Sigma}_{c_i|s_i}, \boldsymbol{\mu}_{c_i|s_i})_{1 \leq i \leq N}$ evolves over time, but for notational convenience, their time index is omitted.

2. Compute model posterior probabilities for all patches: $\forall 1 \leq i \leq N,\ 0 \leq s_i \leq K-1$,

$$\mathbb{P}_t(s_i|\tilde{P}_i) \propto \boldsymbol{w}_{s_i} \exp\Big(\frac{1}{2} \ln\det \boldsymbol{\Sigma}_{c_i|s_i} + \frac{\boldsymbol{\mu}_{c_i|s_i}^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1} \boldsymbol{\mu}_{c_i|s_i}}{2} - \frac{\|\mathfrak{M}_i \boldsymbol{\mu}_{s_i} - \tilde{P}_i\|^2}{2\sigma^2}\Big).$$

under the constraint $\sum_{k=0}^{K-1} \mathbb{P}_t(s_i = k|\tilde{P}_i) = 1$.

Maximization:

1. Update model priors: $\forall 0 \leq k \leq K-1,\ \ \boldsymbol{w}_{k,t} = \frac{1}{N} \sum_{i=1}^N \mathbb{P}_t(s_i = k|\tilde{P}_i)$.
2. Update noise variance:

$$\sigma_t^2 = \frac{\sum_{i=1}^N \sum_{k=0}^{K-1} \mathbb{P}_t(s_i = k|\tilde{P}_i) \int dc_i p_i(c_i|s_i = k) \|\tilde{P}_i - \mathfrak{M}_i \tilde{\boldsymbol{F}}_{s_i,t-1} \tilde{c}_i\|^2}{\sum_{i=1}^N |\mathfrak{M}_i|}$$

where $|\mathfrak{M}_i|$ means the number of non-zero entries in $\mathfrak{M}_i$. See (6.4) for the integral.

3. Update model factors and means: solve the linear equation one row at a time

$$\sum_{i=1}^N \mathfrak{M}_i \tilde{\boldsymbol{F}}_{s_i,t} \mathbb{P}_t(s_i|\tilde{P}_i) \begin{pmatrix} \boldsymbol{\Sigma}_{\boldsymbol{c_i}|\boldsymbol{s_i}} + \boldsymbol{\mu}_{c_i|s_i} \boldsymbol{\mu}_{c_i|s_i}^T & \boldsymbol{\mu}_{c_i|s_i} \\ \boldsymbol{\mu}_{c_i|s_i}^T & 1 \end{pmatrix} = \sum_{i=1}^N \mathfrak{M}_i \mathbb{P}_t(s_i|\tilde{P}_i) \tilde{P}_i \begin{pmatrix} \boldsymbol{\mu}_{c_i|s_i}^T & 1 \end{pmatrix}$$

where $\tilde{\boldsymbol{F}}_{s_i,t} := [\boldsymbol{F}_{s_i,t}, \boldsymbol{\mu}_{s_i,t}]$.

end for
Create the patch map:

$$f : \tilde{P}_i \in \tilde{\mathcal{P}} \mapsto \underset{0 \leq k \leq K-1}{\operatorname{argmax}}\ \mathbb{P}_{\Theta_S}(s_i = k|\tilde{P}_i).$$

Filter: $\forall \tilde{P}_i \in \tilde{\mathcal{P}}$, take the model $k_i = f(\tilde{P}_i)$ and fill in $\tilde{P}_i$'s missing pixel values with the estimates from

$$\widehat{P}_i = \boldsymbol{F}_{k_i,S} \boldsymbol{\beta_i} + \boldsymbol{\mu}_{k_i,S} \text{ with } \boldsymbol{\beta}_i = \xi(I_{l_k} + \xi \boldsymbol{F}_{k_i,S}^T \mathfrak{M}_i \boldsymbol{F}_{k_i,S})^{-1} \boldsymbol{F}_{k_i,S}^T \mathfrak{M}_i (\tilde{P}_i - \boldsymbol{\mu}_{k_i,S}).$$

Assemble: assign equal weights to all restored patches and recover the image in the usual way.

(a) (b) (c) (d)

Figure 6.2 – *test images: (a) parrot (b) shapes (c) barbara (d) frog*

6.3.3 Numerical results

Figure 6.2 shows the images used in our experiments. Table 6.1 compares the results of different inpainting algorithms in RMSE.

Table 6.1 – *Algorithm Comparison in RMSE*

text	barbara	frog	parrot	shapes
EPLL	5.7	4.8	6.4	6.5
PLE	**4.2**	**4.3**	**6.1**	5.9
E-PLE	5.0	4.7	6.6	**5.7**
rand 0.2	barbara	frog	parrot	shapes
EPLL	2.5	2.5	4.0	2.8
PLE	**1.7**	**2.0**	**3.7**	2.4
E-PLE	1.9	2.2	3.8	**2.3**
rand 0.4	barbara	frog	parrot	shapes
EPLL	4.7	4.5	**6.7**	5.6
PLE	**3.7**	**4.0**	6.9	5.4
E-PLE	**3.7**	4.1	**6.7**	**4.7**
rand 0.6	barbara	frog	parrot	shapes
EPLL	8.6	7.2	**9.5**	9.6
PLE	10.6	7.1	10.9	11.2
E-PLE	**7.9**	**6.8**	10.0	**8.8**
rand 0.8	barbara	frog	parrot	shapes
EPLL	**15.8**	11.2	15.0	17.6
PLE	20.1	11.0	16.0	19.4
E-PLE	16.9	**10.8**	**14.8**	**16.5**

Comments:

1. For fairness, all the algorithms used the same masked images. A random mask has a certain fixed probability for each pixel to become invisible. Both PLE and E-PLE iterate six times. And EPLL refers to the algorithm developed in Zoran and Weiss (2011).
2. The higher the masking ratio, the worse the recovery in all cases. A higher masking ratio also implies that an algorithm has to guess more so that a well constructed prior knowledge is the most needed. Lacking such a structure, PLE does not do as well as the other two.
3. For natural images, one single iteration of E-PLE usually suffices to achieve a good restoration (see figure 6.3). More iterations do guarantee an increase in likelihood Dempster et al. (1977), though not

necessarily in RMSE. Yet for a highly degraded image, more iterations could allow better inpainting especially for those images rich in structure such as `barbara`. The same cannot be said of PLE.

4. On the contrary, in case of artificial images, it is desirable to have the algorithm update mixture components through learning in order to adapt itself to this unexpected reality. This explains why EPLL yields a consistently worse result with `shapes` (see figure 6.4).

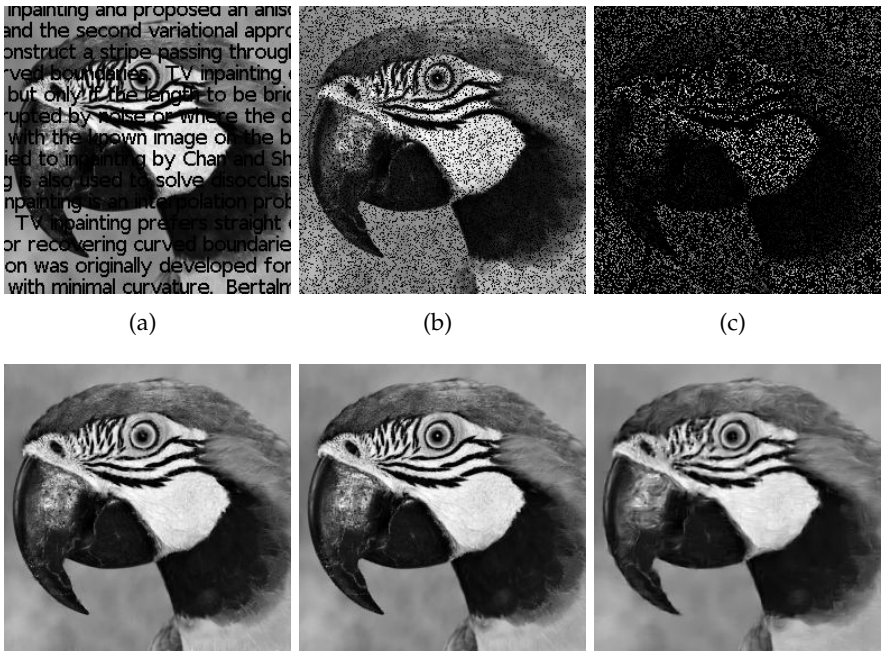
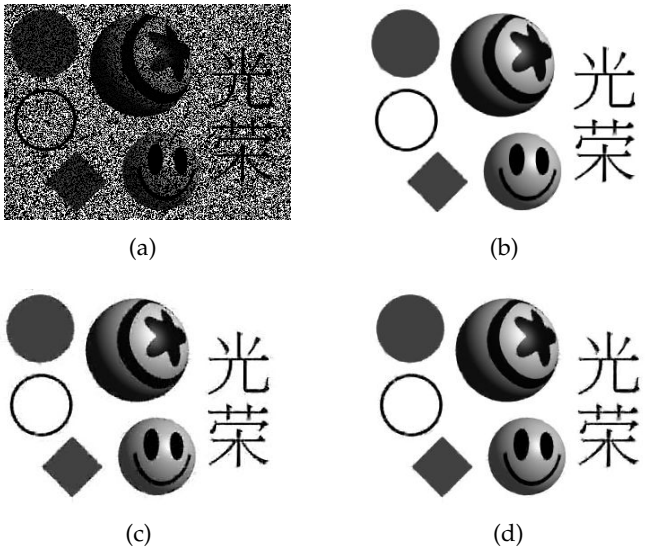


Figure 6.3 – *E-PLE iterates once on the masked images. (a) text masked image (d) inpainted (RMSE = 7.9) (b) randomly masked image (ratio = 0.2) (e) inpainted (RMSE = 5.0) (c) randomly masked image (ratio = 0.8) (f) inpainted (RMSE = 14.9).*



(a) (b) (c) (d)

Figure 6.4 – *Inpainting an artificial image (a) masked* `shapes` *(40% pixels visible) (b) inpainted with EPLL (RMSE = 9.6) (c) inpainted with PLE (six iterations and RMSE = 11.2) (d) inpainted with E-PLE (six iterations and RMSE = 8.8).*

6.4 Appendix

In this section, EM used in E-PLE is derived.
E-PLE's observation model is

$$\tilde{P} = \mathfrak{M}\Big(\sum_{k=0}^{K-1} 1_{s=k} P + N\Big) = \sum_{k=0}^{K-1} \mathfrak{M}(P+N) 1_{s=k}$$

whereby the patch is affected by Gaussian noise N and a mask $\mathfrak{M}$. The patch model s is governed by the mixing weights $\boldsymbol{w}_\cdot$ and independent of N. Let Θ be the parameter set comprising $\big(\boldsymbol{F}_k, \boldsymbol{\mu}_k, \boldsymbol{w}_k\big)_{0\le k\le K-1}$, and the noise standard deviation σ.

Lemme 6.1 *Given the linear model*

$$\forall 1 \le i \le N, \quad \tilde{P}_i = \sum_{k=0}^{K-1} \mathfrak{M}_i(\boldsymbol{F}_k c_i + \boldsymbol{\mu}_k + \sigma n_i) 1_{s_i=k}$$

the posterior law of the coefficient c_i conditional on $(\tilde{P}_i, s_i)$ is Gaussian and its density $p_i(c_i|s_i)$ is characterized by the covariance matrix and mean

$$\boldsymbol{\Sigma}_{c_i|s_i} = \Big(\frac{\boldsymbol{F}_{s_i}^T \mathfrak{M}_i \boldsymbol{F}_{s_i}}{\sigma^2} + I\Big)^{-1} \quad \textit{and} \quad \boldsymbol{\mu}_{c_i|s_i} = \boldsymbol{\Sigma}_{c_i|s_i} \frac{\boldsymbol{F}_{s_i}^T (\tilde{P}_i - \mathfrak{M}_i \boldsymbol{\mu}_{s_i})}{\sigma^2}.$$

Moreover, the density of $\tilde{P}_i$ given s_i is

$$p_\Theta(\tilde{P}_i|s_i) = C_{\mathfrak{M}_i,\sigma^2} \exp\Big(\frac{1}{2}\ln\det\boldsymbol{\Sigma}_{c_i|s_i} + \frac{\boldsymbol{\mu}_{c_i|s_i}^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1} \boldsymbol{\mu}_{c_i|s_i}}{2} - \frac{\|\mathfrak{M}_i \boldsymbol{\mu}_{s_i} - \tilde{P}_i\|^2}{2\sigma^2}\Big)$$

for some positive constant $C_{\mathfrak{M}_i,\sigma^2}$ only depending on $\mathfrak{M}_i$ and σ^2.

Proof: an elementary application of Bayes formula implies

$$
\begin{aligned}
&p_\Theta(\tilde{P}_i|s_i) = \int dc_i p_\Theta(c_i|s_i) p_\Theta(\tilde{P}_i|c_i, s_i) \\
=&\frac{C_{\mathfrak{M}_i,\sigma^2}}{(2\pi)^{l_{s_i}/2}} \int dc_i \exp\Big(-\frac{\|c_i\|^2}{2} - \frac{\|\mathfrak{M}_i(\boldsymbol{F}_{s_i}c_i + \boldsymbol{\mu}_{s_i}) - \tilde{P}_i\|^2}{2\sigma^2}\Big) \\
=&\frac{C_{\mathfrak{M}_i,\sigma^2}}{(2\pi)^{l_{s_i}/2}} \int dc_i \exp\Big(-\frac{(c_i - \boldsymbol{\mu}_{c_i|s_i})^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1}(c_i - \boldsymbol{\mu}_{c_i|s_i})}{2} + \frac{\boldsymbol{\mu}_{c_i|s_i}^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1} \boldsymbol{\mu}_{c_i|s_i}}{2} - \frac{\|\mathfrak{M}_i\boldsymbol{\mu}_{s_i} - \tilde{P}_i\|^2}{2\sigma^2}\Big) \\
=&C_{\mathfrak{M}_i,\sigma^2} \exp\Big(\frac{1}{2}\ln\det\boldsymbol{\Sigma}_{c_i|s_i} + \frac{\boldsymbol{\mu}_{c_i|s_i}^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1} \boldsymbol{\mu}_{c_i|s_i}}{2} - \frac{\|\mathfrak{M}_i\boldsymbol{\mu}_{s_i} - \tilde{P}_i\|^2}{2\sigma^2}\Big)
\end{aligned}
$$

Hence the lemma's claims. □

As a by-product, we find the posterior probability

$$
\begin{aligned}
\mathbb{P}_\Theta(s_i|\tilde{P}_i) &\propto p_\Theta(\tilde{P}_i|s_i)\mathbb{P}_\Theta(s_i) \\
&\propto w_{s_i} \exp\Big(\frac{1}{2}\ln\det\boldsymbol{\Sigma}_{c_i|s_i} + \frac{\boldsymbol{\mu}_{c_i|s_i}^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1} \boldsymbol{\mu}_{c_i|s_i}}{2} - \frac{\|\mathfrak{M}_i\boldsymbol{\mu}_{s_i} - \tilde{P}_i\|^2}{2\sigma^2}\Big).
\end{aligned}
$$

Hence $\tilde{P}_i$ is best associated to

$$
k_i = \underset{0\le s_i\le K-1}{\operatorname{argmax}} \Big(\ln w_{s_i} + \frac{1}{2}\ln\det\boldsymbol{\Sigma}_{c_i|s_i} + \frac{\boldsymbol{\mu}_{c_i|s_i}^T \boldsymbol{\Sigma}_{c_i|s_i}^{-1} \boldsymbol{\mu}_{c_i|s_i}}{2} - \frac{\|\mathfrak{M}_i\boldsymbol{\mu}_{s_i} - \tilde{P}_i\|^2}{2\sigma^2}\Big).
$$

With the parameter set Θ_t known at time t, EM first calculates the conditional expectation of the log-likelihood completed with latent variables $(s_i, c_i)_{1\le i\le N}$ (by abuse of notation, the probabilities $\mathbb{P}$ and densities p are mixed up as the context is clear)

$$
\begin{aligned}
&\sum_{i=1}^N \mathbb{E}_{\Theta_t}\big[\ln\mathbb{P}_\Theta(\tilde{P}_i, s_i, c_i)|\tilde{P}_i\big] \\
=&\sum_{i=1}^N\sum_{k=0}^{K-1} \mathbb{E}_{\Theta_t}\big[\ln\mathbb{P}_\Theta(\tilde{P}_i, s_i, c_i)|\tilde{P}_i, s_i = k\big]\mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \\
=&\sum_{i=1}^N\sum_{k=0}^{K-1} \Big(\mathbb{E}_{\Theta_t}\big[\ln\mathbb{P}_\Theta(\tilde{P}_i, c_i|s_i = k)|\tilde{P}_i, s_i = k\big] + \ln w_k\Big)\mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \\
=&\sum_{i=1}^N\sum_{k=0}^{K-1} \Big(\mathbb{E}_{\Theta_t}\big[-\frac{\|c_i\|^2}{2} - \frac{\|\tilde{P}_i - \mathfrak{M}_i(\boldsymbol{F}_k c_i + \boldsymbol{\mu}_k)\|^2}{2\sigma^2} - \frac{|\mathfrak{M}_i|}{2}\ln\sigma^2 + C_{k,\mathfrak{M}_i}|\tilde{P}_i, s_i = k\big] + \ln w_k\Big)\mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i)
\end{aligned}
$$

where $|\mathfrak{M}_i|$ is the number of non-zero elements in $\mathfrak{M}_i$ and $C_{k,\mathfrak{M}_i}$ is a constant that depends only on the couple $(k, \mathfrak{M}_i)$. The only variables that remain random in the conditional expectation are $(c_i)_{1\le i\le N}$ and this allows us to put the previous lemma to good use. Since only the second order moments are involved, the computation is straightforward:

$$
\begin{aligned}
&\mathbb{E}_{\Theta_t}\big[\|c_i\|^2|\tilde{P}_i, s_i\big] = tr(\boldsymbol{\Sigma}_{\boldsymbol{c_i|s_i}} + \boldsymbol{\mu}_{c_i|s_i}\boldsymbol{\mu}_{c_i|s_i}^T) := tr(\mathcal{C}_i) \qquad (6.4)\\
&\mathbb{E}_{\Theta_t}\big[\|\tilde{P}_i - \mathfrak{M}_i(\boldsymbol{F}_k c_i + \boldsymbol{\mu}_k)\|^2|\tilde{P}_i, s_i\big] = \|\tilde{P}_i - \mathfrak{M}_i\boldsymbol{\mu}_k\|^2 - 2\big\langle\tilde{P}_i - \boldsymbol{\mu}_k, \mathfrak{M}_i\boldsymbol{F}_k\boldsymbol{\mu}_{c_i|s_i}\big\rangle + tr(\mathcal{C}_i\boldsymbol{F}_k^T\mathfrak{M}_i\boldsymbol{F}_k).
\end{aligned}
$$

Next, EM maximizes the expectation just obtained w.r.t. the model parameters. For a more compact expression, let us combine the factor loading matrix $\boldsymbol{F}_{s_i}$ with the mean $\boldsymbol{\mu}_{s_i}$ to form $\tilde{\boldsymbol{F}}_{s_i}$ (thus the coefficient c_i is extended

by one additional constant equal to 1). Now derive the expectation w.r.t. $\tilde{\boldsymbol{F}}_k$ and set it to zero

$$\frac{\partial}{\partial \tilde{\boldsymbol{F}}_k} \sum_{i=1}^{N} \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \int d\tilde{c}_i p_i(\tilde{c}_i|s_i = k) \|\tilde{P}_i - \mathfrak{M}_i \tilde{\boldsymbol{F}}_k \tilde{c}_i\|^2 = 0$$

which leads to

$$\sum_{i=1}^{N} \mathfrak{M}_i \tilde{\boldsymbol{F}}_k \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \int d\tilde{c}_i p_i(\tilde{c}_i|s_i = k) \tilde{c}_i \tilde{c}_i^T = \sum_{i=1}^{N} \mathfrak{M}_i \tilde{P}_i \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \int d\tilde{c}_i p_i(\tilde{c}_i|s_i = k) \tilde{c}_i^T.$$

Updating $(\tilde{\boldsymbol{F}}_k)_{0 \leq k \leq K-1}$ amounts to solving a linear equation: denoting by $(M)_q$ the q-th row of a matrix M, we have $\forall 1 \leq q \leq \kappa^2$

$$(\tilde{\boldsymbol{F}}_k)_q \sum_{i=1}^{n} \delta_{\mathfrak{M}_i(q,q),q} \mathbb{P}_{\Theta_t}(s_i|\tilde{P}_i) \int d\tilde{c}_i p_i(\tilde{c}_i|s_i) \tilde{c}_i \tilde{c}_i^T = \Big(\sum_{i=1}^{N} \mathfrak{M}_i \tilde{P}_i \mathbb{P}_{\Theta_t}(s_i|\tilde{P}_i) \int d\tilde{c}_i p_i(\tilde{c}_i|s_i) \tilde{c}_i^T \Big)_q$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta and

$$\int d\tilde{c}_i p_i(\tilde{c}_i|s_i) \tilde{c}_i \tilde{c}_i^T = \begin{pmatrix} \mathcal{C}_i & \boldsymbol{\mu}_{c_i|s_i} \\ \boldsymbol{\mu}_{c_i|s_i}^T & 1 \end{pmatrix}, \quad \int d\tilde{c}_i p_i(\tilde{c}_i|s_i) \tilde{c}_i^T = \begin{pmatrix} \boldsymbol{\mu}_{c_i|s_i}^T & 1 \end{pmatrix}.$$

Hence, if none of the observed patches has a visible pixel at row q, we will not be able to estimate the factors' or means' coordinate at that position. However, it rarely happens if we have a large enough dataset and that the mask behaves sufficiently randomly.

Similarly, the new model prior can be found via the next problem

$$\underset{\boldsymbol{w}_1, \cdots \boldsymbol{w}_{K-1}}{\operatorname{argmax}} \sum_{k=0}^{K-1} \ln \boldsymbol{w}_k \sum_{i=1}^{N} \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \;\; \text{s.t.} \;\; \min_{0 \leq k \leq K-1} \boldsymbol{w}_k \geq 0 \text{ and } \sum_{k=0}^{K-1} \boldsymbol{w}_k = 1$$

whose solution is

$$\forall 0 \leq k \leq K-1, \;\; \boldsymbol{w}_k = \frac{1}{N} \sum_{i=1}^{N} \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i).$$

Finally, the noise level can be estimated by

$$\frac{\partial}{\partial \sigma^2} \sum_{i=1}^{N} \sum_{k=0}^{K-1} \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \int dc_i p_i(c_i|s_i = k) \Big(- \frac{|\mathfrak{M}_i|}{2} \ln \sigma^2 - \frac{\|\tilde{P}_i - \mathfrak{M}_i \tilde{\boldsymbol{F}}_{s_i} \tilde{c}_i\|^2}{2\sigma^2} \Big) = 0$$

whose solution is quite intuitive:

$$\sigma^2 = \frac{\sum_{i=1}^{N} \sum_{k=0}^{K-1} \mathbb{P}_{\Theta_t}(s_i = k|\tilde{P}_i) \int dc_i p_i(c_i|s_i = k) \|\tilde{P}_i - \mathfrak{M}_i \tilde{\boldsymbol{F}}_{s_i} \tilde{c}_i\|^2}{\sum_{i=1}^{N} |\mathfrak{M}_i|}.$$

where the integral is the same as (6.4).

7 An analysis of the Viola-Jones face detection algorithm

In this chapter, we decipher the Viola-Jones algorithm, the first ever real-time face detection system. There are three ingredients working in concert to enable a fast and accurate detection: the integral image for feature computation, Adaboost for feature selection and an attentional cascade for efficient computational resource allocation. Here we propose a complete algorithmic description, a learning code and a learned face detector that can be applied to any color image. Since the Viola-Jones algorithm typically gives multiple detections, a post-processing step is also proposed to reduce detection redundancy using a robustness argument.

7.1 Introduction

A face detector has to tell whether an image of arbitrary size contains a human face and if so, where it is. One natural framework for considering this problem is that of binary classification, in which a classifier is constructed to minimize the misclassification risk. Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance.

This task requires an accurate numerical description of what sets human faces apart from other objects. It turns out that these characteristics can be extracted with a remarkable committee learning algorithm called Adaboost, which relies on a committee of weak classifiers to form a strong one through a voting mechanism. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms.

An operational algorithm must also work with a reasonable computational budget. Techniques such as integral image and attentional cascade make the Viola-Jones algorithm Viola and Jones (2004) highly efficient: fed with a real time image sequence generated from a standard webcam, it performs well on a standard PC.

7.2 Algorithm

To study the algorithm in detail, we start with the image features for the classification task.

7.2.1 Features and integral image

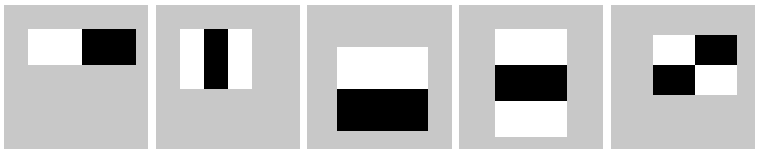
The Viola-Jones algorithm uses Haar-like features, that is, a scalar product between the image and some Haar-like templates. More precisely, let I and P denote an image and a pattern, both of the same size $N \times N$ (see Figure 7.1). The feature associated with pattern P of image I is defined by

$$\sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i,j)1_{P(i,j) \text{ is white}} - \sum_{1\leq i\leq N}\sum_{1\leq j\leq N} I(i,j)1_{P(i,j) \text{ is black}}.$$

To compensate the effect of different lighting conditions, all the images should be mean and variance normalized beforehand. Those images with variance lower than one, having little information of interest in the first place, are left out of consideration.

(a) (b) (c)

Figure 7.1 – *Haar-like features. Here as well as below, the background of a template like (b) is painted gray to highlight the pattern's support. Only those pixels marked in black or white are used when the corresponding feature is calculated.*



(a) (b) (c) (d) (e)

Figure 7.2 – *Five Haar-like patterns. The size and position of a pattern's support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable: a* 24×24 *image, for instance, has* 43200, 27600, 43200, 27600 *and* 20736 *features of category (a), (b), (c), (d) and (e) respectively, hence* 162336 *features in all.*

In practice, five patterns are considered (see Figure 7.2 and Algorithm 11). The derived features are assumed to hold all the information needed to characterize a face. Since faces are by and large regular by nature, the use of Haar-like patterns seems justified. There is, however, another crucial element which lets this set of features take precedence: the integral image which allows to calculate them at a very low computational cost. Instead of summing up all the pixels inside a rectangular window,

Algorithm 11 Computing a 24×24 image's Haar-like feature vector

Input: a 24×24 image with zero mean and unit variance
Output: a $d \times 1$ scalar vector with its feature index $\mathfrak{f}$ ranging from 1 to d
Set the feature index $\mathfrak{f} \leftarrow 0$
Compute feature type (a)
for all (i, j) such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  for all (w, h) such that $i + h - 1 \leq 24$ and $j + 2w - 1 \leq 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    record this feature parametrized by $(1, i, j, w, h)$: $S_1 - S_2$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  end for
end for
Compute feature type (b)
for all (i, j) such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  for all (w, h) such that $i + h - 1 \leq 24$ and $j + 3w - 1 \leq 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + 2w, j + 3w - 1]$
    record this feature parametrized by $(2, i, j, w, h)$: $S_1 - S_2 + S_3$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  end for
end for
Compute feature type (c)
for all (i, j) such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  for all (w, h) such that $i + 2h - 1 \leq 24$ and $j + w - 1 \leq 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    record this feature parametrized by $(3, i, j, w, h)$: $S_1 - S_2$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  end for
end for
Compute feature type (d)
for all (i, j) such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  for all (w, h) such that $i + 3h - 1 \leq 24$ and $j + w - 1 \leq 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    compute the sum S_3 of the pixels in $[i + 2h, i + 3h - 1] \times [j, j + w - 1]$
    record this feature parametrized by $(4, i, j, w, h)$: $S_1 - S_2 + S_3$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  end for
end for
Compute feature type (e)
for all (i, j) such that $1 \leq i \leq 24$ and $1 \leq j \leq 24$ **do**
  for all (w, h) such that $i + 2h - 1 \leq 24$ and $j + 2w - 1 \leq 24$ **do**
    compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
    compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
    compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
    compute the sum S_4 of the pixels in $[i + h, i + 2h - 1] \times [j + w, j + 2w - 1]$
    record this feature parametrized by $(5, i, j, w, h)$: $S_1 - S_2 - S_3 + S_4$
    $\mathfrak{f} \leftarrow \mathfrak{f} + 1$
  end for
end for

this technique mirrors the use of cumulative distribution functions. The integral image I of I

$$I(i,j) := \begin{cases} \sum_{1\le s\le i}\sum_{1\le t\le j} I(s,t), & 1 \le i \le N \text{ and } 1 \le j \le N \\ 0, & \text{otherwise} \end{cases},$$

is so defined that

$$\begin{aligned} &\sum_{N_1\le i\le N_2}\sum_{N_3\le j\le N_4} I(i,j) \\ =&I(N_2,N_4) - I(N_2,N_3-1) - I(N_1-1,N_4) + I(N_1-1,N_3-1), \end{aligned} \quad (7.1)$$

holds for all $N_1 \le N_2$ and $N_3 \le N_4$. As a result, computing an image's rectangular local sum requires at most four elementary operations given its integral image. Moreover, obtaining the integral image itself can be done in linear time: setting $N_1 = N_2$ and $N_3 = N_4$ in (7.1), we find

$$I(N_1,N_3) = I(N_1,N_3) - I(N_1,N_3-1) - I(N_1-1,N_3) + I(N_1-1,N_3-1).$$

Hence a recursive relation which leads to Algorithm 12.

Algorithm 12 Integral Image

1: **Input:** an image I of size $N \times M$.
2: **Output:** its integral image I of the same size.
3: Set $I(1,1) = I(1,1)$.
4: **for** $i = 1$ to N **do**
5: **for** $j = 1$ to M **do**
6: $I(i,j) = I(i,j) + I(i,j-1) + I(i-1,j) - I(i-1,j-1)$ and I is defined to be zero whenever its argument (i,j) ventures out of I's domain.
7: **end for**
8: **end for**

As a side note, let us mention that once the useful features have been selected by the boosting algorithm, one needs to scale them up accordingly when dealing with a bigger window (see Algorithm 13). Smaller windows, however, will not be looked at.

7.2.2 Feature selection with adaboost

How to make sense of these features is the focus of Adaboost Freund and Schapire (1997).

Some terminology. A classifier maps an observation to a label valued in a finite set. For face detection, it assumes the form of $f : \mathbb{R}^d \mapsto \{-1, 1\}$, where 1 means that there is a face and -1 the contrary (see Figure 7.3) and d is the number of Haar-like features extracted from an image. Given the probabilistic weights $w_\cdot \in \mathbb{R}_+$ assigned to a training set made up of n observation-label pairs (x_i, y_i), Adaboost aims to iteratively drive down an upper bound of the empirical loss

$$\sum_{i=1}^{n} w_i 1_{y_i \ne f(x_i)},$$

Algorithm 13 Feature Scaling

Input: an $e \times e$ image with zero mean and unit variance ($e \geq 24$)
Parameter: a Haar-like feature type and its parameter (i, j, w, h) as defined in Algorithm 11
Output: the feature value
if *feature type* (a) **then**
  set the original feature support size $a \leftarrow 2wh$
  $i \leftarrow [\![ie/24]\!], j \leftarrow [\![je/24]\!], h \leftarrow [\![he/24]\!]$ where $[\![z]\!]$ defines the nearest integer to $z \in \mathbb{R}_+$
  $w \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq [\![1 + 2we/24]\!]/2,\ 2\kappa \leq e - j + 1\}$
  compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
  compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
  return the scaled feature $\frac{(S_1 - S_2)a}{2wh}$
end if
if *feature type* (b) **then**
  set the original feature support size $a \leftarrow 3wh$
  $i \leftarrow [\![ie/24]\!], j \leftarrow [\![je/24]\!], h \leftarrow [\![he/24]\!]$
  $w \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq [\![1 + 3we/24]\!]/3,\ 3\kappa \leq e - j + 1\}$
  compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
  compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
  compute the sum S_2 of the pixels in $[i, i + h - 1] \times [j + 2w, j + 3w - 1]$
  return the scaled feature $\frac{(S_1 - S_2 + S_3)a}{3wh}$
end if
if *feature type* (c) **then**
  set the original feature support size $a \leftarrow 2wh$
  $i \leftarrow [\![ie/24]\!], j \leftarrow [\![je/24]\!], w \leftarrow [\![we/24]\!]$
  $h \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq [\![1 + 2he/24]\!]/2,\ 2\kappa \leq e - i + 1\}$
  compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
  compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
  return the scaled feature $\frac{(S_1 - S_2)a}{2wh}$
end if
if *feature type* (d) **then**
  set the original feature support size $a \leftarrow 3wh$
  $i \leftarrow [\![ie/24]\!], j \leftarrow [\![je/24]\!], w \leftarrow [\![we/24]\!]$
  $h \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq [\![1 + 3he/24]\!]/3,\ 3\kappa \leq e - i + 1\}$
  compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
  compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
  compute the sum S_3 of the pixels in $[i + 2h, i + 3h - 1] \times [j, j + w - 1]$
  return the scaled feature $\frac{(S_1 - S_2 + S_3)a}{3wh}$
end if
if *feature type* (e) **then**
  set the original feature support size $a \leftarrow 4wh$
  $i \leftarrow [\![ie/24]\!], j \leftarrow [\![je/24]\!]$
  $w \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq [\![1 + 2we/24]\!]/2,\ 2\kappa \leq e - j + 1\}$
  $h \leftarrow \max\{\kappa \in \mathbb{N} : \kappa \leq [\![1 + 2he/24]\!]/2,\ 2\kappa \leq e - i + 1\}$
  compute the sum S_1 of the pixels in $[i, i + h - 1] \times [j, j + w - 1]$
  compute the sum S_2 of the pixels in $[i + h, i + 2h - 1] \times [j, j + w - 1]$
  compute the sum S_3 of the pixels in $[i, i + h - 1] \times [j + w, j + 2w - 1]$
  compute the sum S_4 of the pixels in $[i + h, i + 2h - 1] \times [j + w, j + 2w - 1]$
  return the scaled feature $\frac{(S_1 - S_2 - S_3 + S_4)a}{4wh}$
end if

under mild technical conditions (see Appendix 7.4). Remarkably, the decision rule constructed by Adaboost remains reasonably simple so that it is not prone to overfitting, which means that the empirically learned rule often generalizes well. For more details on the method, we refer to Freund et al. (1999), Friedman et al. (2001). Despite its groundbreaking success, it ought to be said that Adaboost does not learn what a face should look like all by itself because it is humans, rather than the algorithm, who perform the labeling and the first round of feature selection, as described in the previous section.

(a) (b)

Figure 7.3 – *Some supervised examples: (a) positive examples (b) negative examples. All of them are* 24×24 *grayscale images. See Section 7.2.4 for more on this dataset.*

The building block of the Viola-Jones face detector is a *decision stump*, or a depth one decision tree, parametrized by a feature $\mathfrak{f} \in \{1, \cdots, d\}$, a threshold $\mathfrak{t} \in \mathbb{R}$ and a toggle $\mathcal{T} \in \{-1, 1\}$. Given an observation $x \in \mathbb{R}^d$, a decision stump h predicts its label using the following rule

$$h(x) = (1_{\pi_{\mathfrak{f}} x \geq \mathfrak{t}} - 1_{\pi_{\mathfrak{f}} x < \mathfrak{t}})\mathcal{T} \in \{-1, 1\}, \tag{7.2}$$

where $\pi_{\mathfrak{f}} x$ is the feature vector's $\mathfrak{f}$-th coordinate. Several comments follow:

1. Any additional pattern produced by permuting black and white rectangles in an existing pattern (see Figure 7.2) is superfluous. Because such a feature is merely the opposite of an existing feature, only a sign change for $\mathfrak{t}$ and $\mathcal{T}$ is needed to have the same classification rule.
2. If the training examples are sorted in ascending order of a given feature $\mathfrak{f}$, a linear time exhaustive search on the threshold and toggle can find a decision stump using this feature that attains the lowest empirical loss

$$\sum_{i=1}^{n} w_i 1_{y_i \neq h(x_i)}, \tag{7.3}$$

on the training set (see Algorithm 14). Imagine a threshold placed somewhere on the real line, if the toggle is set to 1, the resulting rule will declare an example x positive if $\pi_{\mathfrak{f}} x$ is greater than the threshold and negative otherwise. This allows us to evaluate the rule's empirical error, thereby selecting the toggle that fits the dataset better (lines 8–16 of Algorithm 14).

Since margin

$$\min_{i:\, y_i=-1} |\pi_f x_i - t| + \min_{i:\, y_i=1} |\pi_f x_i - t|,$$

and risk, or the expectation of the empirical loss (7.3), are closely related Friedman et al. (2001), Rosenblatt (1958), Schapire et al. (1998), of two decision stumps having the same empirical risk, the one with a larger margin is preferred (line 14 of Algorithm 14). Thus in the absence of duplicates, there are $n + 1$ possible thresholds and the one with the smallest empirical loss should be chosen. However it is possible to have the same feature values from different examples and extra care must be taken to handle this case properly (lines 27–32 of Algorithm 14).

By adjusting individual example weights (Algorithm 16 line 10), Adaboost makes more effort to learn harder examples and adds more decision stumps (see Algorithm 15) in the process. Intuitively, in the final voting, a stump h_t with lower empirical loss is rewarded with a bigger say (a higher α_t, see Algorithm 16 line 9) when a T-member committee (vote-based classifier) assigns an example according to

$$f^T(\cdot) = \text{sign}\,\big[\sum_{t=1}^{T} \alpha_t h_t(\cdot)\big].$$

How the training examples should be weighed is explained in detail in Appendix 7.4. Figure 7.4 shows an instance where Adaboost reduces false positive and false negative rates simultaneously as more and more stumps are added to the committee. For notational simplicity, we denote the empirical loss by

$$\sum_{i=1}^{n} w_i(1) 1_{y_i \sum_{t=1}^{T} \alpha_t h_t(x_i) \le 0} := \mathbb{P}(f^T(X) \neq Y),$$

where (X, Y) is a random couple distributed according to the probability $\mathbb{P}$ defined by the weights $w_i(1)$, $1 \le i \le n$ set when the training starts. As the empirical loss goes to zero with T, so do both false positive $\mathbb{P}(f^T(X) = 1|Y = -1)$ and false negative rates $\mathbb{P}(f^T(X) = -1|Y = 1)$ owing to

$$\mathbb{P}(f^T(X) \neq Y) = \mathbb{P}(Y = 1)\mathbb{P}(f^T(X) = -1|Y = 1) + \mathbb{P}(Y = -1)\mathbb{P}(f^T(X) = 1|Y = -1).$$

Thus the detection rate

$$\mathbb{P}(f^T(X) = 1|Y = 1) = 1 - \mathbb{P}(f^T(X) = -1|Y = 1),$$

must tend to 1.

Thus the size T of the trained committee depends on the targeted false positive and false negative rates. In addition, let us mention that, given n_- negative and n_+ positive examples in a training pool, it is customary to give a negative (resp. positive) example an initial weight equal to $0.5/n_-$ (resp. $0.5/n_+$) so that Adaboost does not favor either category at the beginning.

Algorithm 14 Decision Stump by Exhaustive Search

Input: n training examples arranged in ascending order of feature $\pi_{\mathfrak{f}} x_i$: $\pi_{\mathfrak{f}} x_{i_1} \leq \pi_{\mathfrak{f}} x_{i_2} \leq \cdots \leq \pi_{\mathfrak{f}} x_{i_n}$, probabilistic example weights $(w_k)_{1 \leq k \leq n}$.
Output: the decision stump's threshold τ, toggle $\mathcal{T}$, error $\mathcal{E}$ and margin $\mathcal{M}$.
Initialization: $\tau \leftarrow \min_{1 \leq i \leq n} \pi_{\mathfrak{f}} x_i - 1$, $\mathcal{M} \leftarrow 0$ and $\mathcal{E} \leftarrow 2$ (an arbitrary upper bound of the empirical loss).
Sum up the weights of the positive (resp. negative) examples whose $\mathfrak{f}$-th feature is bigger than the present threshold: $W_1^+ \leftarrow \sum_{i=1}^n w_i 1_{y_i=1}$ (resp. $W_{-1}^+ \leftarrow \sum_{i=1}^n w_i 1_{y_i=-1}$).
Sum up the weights of the positive (resp. negative) examples whose $\mathfrak{f}$-th feature is smaller than the present threshold: $W_1^- \leftarrow 0$ (resp. $W_{-1}^- \leftarrow 0$).
Set iterator $j \leftarrow 0$, $\widehat{\tau} \leftarrow \tau$ and $\widehat{\mathcal{M}} \leftarrow \mathcal{M}$.
while true **do**
  Select the toggle to minimize the weighted error: $\text{error}_+ \leftarrow W_1^- + W_{-1}^+$ and $\text{error}_- \leftarrow W_1^+ + W_{-1}^-$.
  if $\text{error}_+ < \text{error}_-$ **then**
    $\widehat{\mathcal{E}} \leftarrow \text{error}_+$ and $\widehat{\mathcal{T}} \leftarrow 1$.
  else
    $\widehat{\mathcal{E}} \leftarrow \text{error}_-$ and $\widehat{\mathcal{T}} \leftarrow -1$.
  end if
  if $\widehat{\mathcal{E}} < \mathcal{E}$ or $\widehat{\mathcal{E}} = \mathcal{E}$ & $\widehat{\mathcal{M}} > \mathcal{M}$ **then**
    $\mathcal{E} \leftarrow \widehat{\mathcal{E}}$, $\tau \leftarrow \widehat{\tau}$, $\mathcal{M} \leftarrow \widehat{\mathcal{M}}$ and $\mathcal{T} \leftarrow \widehat{\mathcal{T}}$.
  end if
  if $j = n$ **then**
    Break.
  end if
  $j \leftarrow j + 1$.
  while true **do**
    if $y_{i_j} = -1$ **then**
      $W_{-1}^- \leftarrow W_{-1}^- + w_{i_j}$ and $W_{-1}^+ \leftarrow W_{-1}^+ - w_{i_j}$.
    else
      $W_1^- \leftarrow W_1^- + w_{i_j}$ and $W_1^+ \leftarrow W_1^+ - w_{i_j}$.
    end if
    To find a new valid threshold, we need to handle duplicate features.
    if $j = n$ or $\pi_{\mathfrak{f}} x_{i_j} \neq \pi_{\mathfrak{f}} x_{i_{j+1}}$ **then**
      Break.
    else
      $j \leftarrow j + 1$.
    end if
  end while
  if $j = n$ **then**
    $\widehat{\tau} \leftarrow \max_{1 \leq i \leq n} \pi_{\mathfrak{f}} x_i + 1$ and $\widehat{\mathcal{M}} \leftarrow 0$.
  else
    $\widehat{\tau} \leftarrow (\pi_{\mathfrak{f}} x_{i_j} + \pi_{\mathfrak{f}} x_{i_{j+1}})/2$ and $\widehat{\mathcal{M}} \leftarrow \pi_{\mathfrak{f}} x_{i_{j+1}} - \pi_{\mathfrak{f}} x_{i_j}$.
  end if
end while

Algorithm 15 Best Stump

1: **Input:** n training examples, their probabilistic weights $(w_i)_{1\leq i\leq n}$, number of features d.
2: **Output:** the best decision stump's threshold, toggle, error and margin.
3: Set the best decision stump's error to 2.
4: **for** $\mathfrak{f} = 1$ to d **do**
5: Compute the decision stump associated with feature $\mathfrak{f}$ using Algorithm 14.
6: **if** this decision stump has a lower weighted error (7.3) than the best stump or a wider margin if the weighted error are the same **then**
7: set this decision stump to be the best.
8: **end if**
9: **end for**

Algorithm 16 Adaboost

1: **Input:** n training examples $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\},\ 1 \leq i \leq n$, number of training rounds T.
2: **Parameter:** the initial probabilistic weights $w_i(1)$ for $1 \leq i \leq n$.
3: **Output:** a strong learner/committee.
4: **for** $t = 1$ to T **do**
5: Run Algorithm 15 to train a decision stump h_t using the weights $w_\cdot(t)$ and get its weighted error ϵ_t

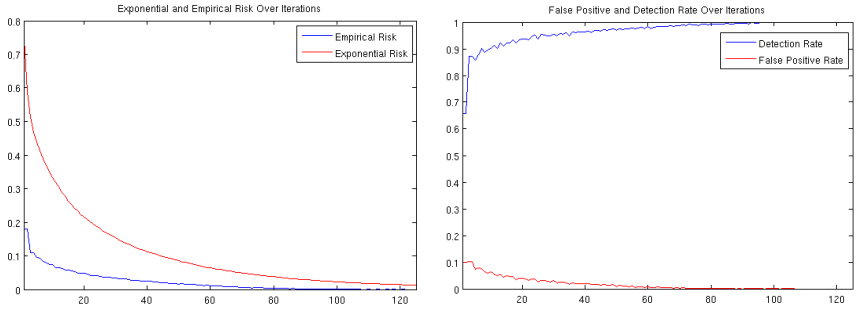
$$\epsilon_t = \sum_{i=1}^{n} w_i(t) 1_{h_t(x_i)\neq y_i},$$

6: **if** $\epsilon_t = 0$ and $t = 1$ **then**
7: training ends and return $h_1(\cdot)$.
8: **else**
9: set $\alpha_t = \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$.
10: update the weights

$$\forall i,\ \ w_i(t+1) = \frac{w_i(t)}{2}\Big(\frac{1}{\epsilon_t} 1_{h_t(x_i)\neq y_i} + \frac{1}{1-\epsilon_t} 1_{h_t(x_i)=y_i}\Big).$$

11: **end if**
12: **end for**
13: Return the rule

$$f^T(\cdot) = \text{sign}\,\Big[\sum_{t=1}^{T} \alpha_t h_t(\cdot)\Big].$$



(a) (b)

Figure 7.4 – *Algorithm 16 ran with equally weighted* 2500 *positive and* 2500 *negative examples. Figure (a) shows that the empirical risk and its upper bound, interpreted as the exponential loss (see Appendix 7.4), decrease steadily over iterations. This implies that false positive and false negative rates must also decrease, as observed in (b).*

7.2.3 Attentional cascade

In theory, Adaboost can produce a single committee of decision stumps that generalizes well. However, to achieve that, an enormous negative training set is needed at the outset to gather all possible negative patterns. In addition, a single committee implies that all the windows inside an image have to go through the same lengthy decision process. There has to be another more cost-efficient way.

The prior probability for a face to appear in an image bears little relevance to the presented classifier construction because it requires both the empirical false negative and false positive rate to approach zero. However, our own experience tells us that in an image, a rather limited number of sub-windows deserve more attention than others. This is true even for face-intensive group photos. Hence the idea of a multi-layer attentional cascade which embodies a principle akin to that of Shannon coding: the algorithm should deploy more resources to work on those windows more likely to contain a face while spending as little effort as possible on the rest.

Each layer in the attentional cascade is expected to meet a training target expressed in false positive and false negative rates: among n negative examples declared positive by all of its preceding layers, layer l ought to recognize at least $(1-\gamma_l)n$ as negative and meanwhile try not to sacrifice its performance on the positives: the detection rate should be maintained above $1-\beta_l$.

At the end of the day, only the generalization error counts which unfortunately can only be estimated with some validation examples that Adaboost is not allowed to see at the training phase. Hence in Algorithm 20 at line 10, a conservative choice is made as to how one assesses the error rates: the higher false positive rate obtained from training and validation is used to evaluate how well the algorithm has learned to distinguish faces from non-faces. The false negative rate is assessed in the same way.

It should be kept in mind that Adaboost by itself does not favor either error rate: it aims to reduce both simultaneously rather than one at the expense of the other. To allow flexibility, one additional control $\mathfrak{s} \in [-1, 1]$

is introduced to shift the classifier

$$f_{\mathfrak{s}}^{T}(\cdot) = \operatorname{sign}\big[\sum_{t=1}^{T} \alpha_t \big(h_t(\cdot) + \mathfrak{s}\big)\big], \tag{7.4}$$

so that a strictly positive $\mathfrak{s}$ makes the classifier more inclined to predict a face and vice versa.

To enforce an efficient resource allocation, the committee size should be small in the first few layers and then grow gradually so that a large number of easy negative patterns can be eliminated with little computational effort (see Figure 7.5).

Algorithm 17 Detecting faces with an Adaboost trained cascade classifier

1: **Input:** an $M \times N$ grayscale image I and an L-layer cascade of shifted classifiers trained using Algorithm 20
2: **Parameter:** a window scale multiplier $\mathfrak{c}$
3: **Output:** $\mathcal{P}$, the set of windows declared positive by the cascade
4: Set $\mathcal{P} = \{[i, i+e-1] \times [j, j+e-1] \subset I : e = [\![24\mathfrak{c}^{\kappa}]\!], \kappa \in \mathbb{N}\}$
5: **for** $l = 1$ to L **do**
6: **for** every window in $\mathcal{P}$ **do**
7: Remove the windowed image's mean and compute its standard deviation.
8: **if** the standard deviation is bigger than 1 **then**
9: divide the image by this standard deviation and compute its features required by the shifted classifier at layer l with Algorithm 13
10: **if** the cascade's l-th layer predicts negative **then**
11: discard this window from $\mathcal{P}$
12: **end if**
13: **else**
14: discard this window from $\mathcal{P}$
15: **end if**
16: **end for**
17: **end for**
18: Return $\mathcal{P}$

Appending a layer to the cascade means that the algorithm has learned to reject a few new negative patterns previously viewed as difficult, all the while keeping more or less the same positive training pool. To build the next layer, more negative examples are thus required to make the training process meaningful. To replace the detected negatives, we run the cascade on a large set of gray images with no human face and collect their false positive windows. The same procedure is used for constructing and replenishing the validation set (see Algorithm 17). Since only 24×24 sized examples can be used in the training phase, those bigger false positives are down-sampled (Algorithm 18) and recycled using Algorithm 19.

Assume that at layer l, a committee of T_l weak classifiers is formed along with a shift $\mathfrak{s}$ so that the classifier's performance on training and validation set can be measured. Let us denote the achieved false positive and false negative rate by $\widehat{\gamma}_l$ and $\widehat{\beta}_l$. Depending on their relation with the targets γ_l and β_l, four cases are presented:

1. If the layer training target is fulfilled (Algorithm 20 line 11: $\widehat{\gamma}_l \leq \gamma_l$ and $\widehat{\beta}_l \leq \beta_l$), the algorithm moves on to training the next layer if necessary.

Algorithm 18 Downsampling a square image

Input: an $e \times e$ image I ($e > 24$)
Output: a downsampled image O of dimension 24×24
Blur I using a Gaussian kernel with standard deviation $\sigma = 0.6\sqrt{(\frac{e}{24})^2 - 1}$
Allocate a matrix O of dimension 24×24
for $i = 0$ to 23 **do**
  for $j = 0$ to 23 **do**
    Compute the scaled coordinates $\tilde{i} \leftarrow \frac{e-1}{25}(i+1)$, $\tilde{j} \leftarrow \frac{e-1}{25}(j+1)$
    Set $\tilde{i}_{max} \leftarrow \min(\llbracket \tilde{i} \rrbracket + 1, e-1)$, $\tilde{i}_{min} \leftarrow \max(0, \llbracket \tilde{i} \rrbracket)$, $\tilde{j}_{max} \leftarrow \min(\llbracket \tilde{j} \rrbracket + 1, e-1)$, $\tilde{j}_{min} \leftarrow \max(0, \llbracket \tilde{j} \rrbracket)$
    Set $O(i,j) = \frac{1}{4}\left[I(\tilde{i}_{max}, \tilde{j}_{max}) + I(\tilde{i}_{min}, \tilde{j}_{max}) + I(\tilde{i}_{min}, \tilde{j}_{min}) + I(\tilde{i}_{max}, \tilde{j}_{min})\right]$
  end for
end for
Return O

Algorithm 19 Collecting false positive examples for training a cascade's $(L+1)$-th layer

Input: a set of grayscale images with no human faces and an L-layer cascade of shifted classifiers
Parameter: a window scale multiplier $\mathfrak{c}$
Output: a set of false positive examples $\mathcal{V}$
for every grayscale image **do**
  Run Algorithm 17 to get all of its false positives $\mathcal{Q}$
  for every windowed image in $\mathcal{Q}$ **do**
    if the window size is bigger than 24×24 **then**
      downsample this subimage using Algorithm 18 and run Algorithm 17 on it
      if the downsampled image remains positive **then**
        accept this false positive to $\mathcal{V}$
      end if
    else
      accept this false positive to $\mathcal{V}$
    end if
  end for
end for
Return $\mathcal{V}$

Algorithm 20 Attentional Cascade

1: **Input:** n training positives, m validation positives, two sets of gray images with no human faces to draw training and validation negatives, desired overall false positive rate γ_o, and targeted layer false positive and detection rate γ_l and $1-\beta_l$.
2: **Parameter:** maximum committee size at layer l: $N_l = \min(10l+10, 200)$.
3: **Output:** a cascade of committees.
4: Set the attained overall false positive rate $\widehat{\gamma}_o \leftarrow 1$ and layer count $l \leftarrow 0$.
5: Randomly draw $10n$ negative training examples and m negative validation examples.
6: **while** $\widehat{\gamma}_o > \gamma_o$ **do**
7: $u \leftarrow 10^{-2}$, $l \leftarrow l+1$, $\mathfrak{s}_l \leftarrow 0$, and $T_l \leftarrow 1$.
8: Run Algorithm 16 on the training set to produce a classifier $f_l^{T_l} = \text{sign}\left[\sum_{t=1}^{T_l} \alpha_t h_t\right]$.
9: Run the $\mathfrak{s}_l$-shifted classifier $f_{l,\mathfrak{s}_l}^{T_l} = \text{sign}\left[\sum_{t=1}^{T_l} \alpha_t (h_t + \mathfrak{s}_l)\right]$, on both the training and validation set to obtain the empirical and generalized false positive (resp. false negative) rate γ_e and γ_g (resp. β_e and β_g).
10: $\widehat{\gamma}_l \leftarrow \max(\gamma_e, \gamma_g)$ and $\widehat{\beta}_l \leftarrow \max(\beta_e, \beta_g)$.
11: **if** $\widehat{\gamma}_l \le \gamma_l$ and $1-\widehat{\beta}_l \ge 1-\beta_l$ **then**
12: $\widehat{\gamma}_o \leftarrow \widehat{\gamma}_o \times \widehat{\gamma}_l$.
13: **else if** $\widehat{\gamma}_l \le \gamma_l$, $1-\widehat{\beta}_l < 1-\beta_l$ and $u > 10^{-5}$ (there is room to improve the detection rate) **then**
14: $\mathfrak{s}_l \leftarrow \mathfrak{s}_l + u$.
15: **if** the trajectory of $\mathfrak{s}_l$ is not monotone **then**
16: $u \leftarrow u/2$.
17: $\mathfrak{s}_l \leftarrow \mathfrak{s}_l - u$.
18: **end if**
19: Go to line 9.
20: **else if** $\widehat{\gamma}_l > \gamma_l$, $1-\widehat{\beta}_l \ge 1-\beta_l$ and $u > 10^{-5}$ (there is room to improve the false positive rate) **then**
21: $\mathfrak{s}_l \leftarrow \mathfrak{s}_l - u$.
22: **if** the trajectory of $\mathfrak{s}_l$ is not monotone **then**
23: $u \leftarrow u/2$.
24: $\mathfrak{s}_l \leftarrow \mathfrak{s}_l + u$.
25: **end if**
26: Go to line 9.
27: **else**
28: **if** $T_l > N_l$ **then**
29: $\mathfrak{s}_l \leftarrow -1$
30: **while** $1-\widehat{\beta}_l < 0.99$ **do**
31: Run line 9 and 10.
32: **end while**
33: $\widehat{\gamma}_o \leftarrow \widehat{\gamma}_o \times \widehat{\gamma}_l$.
34: **else**
35: $T_l \leftarrow T_l + 1$ (Train one more member to add to the committee.)
36: Go to line 8.
37: **end if**
38: **end if**
39: Remove the false negatives and true negatives detected by the current cascade

$$f_{cascade}(X) = 2\Big(\prod_{p=1}^{l} 1_{f_{p,\mathfrak{s}_p}^{T_p}(X)=1} - \frac{1}{2}\Big).$$

Use this cascade with Algorithm 19 to draw some false positives so that there are n training negatives and m validation negatives for the next round.
40: **end while**
41: Return the cascade.

2. If there is room to improve the detection rate (Algorithm 20 line 13: $\widehat{\gamma}_l \leq \gamma_l$ and $\widehat{\beta}_l > \beta_l$), $\mathfrak{s}$ is increased by u, a prefixed unit.
3. If there is room to improve the false positive rate (Algorithm 20 line 20: $\widehat{\gamma}_l > \gamma_l$ and $\widehat{\beta}_l \leq \beta_l$), $\mathfrak{s}$ is decreased by u.
4. If both error rates fall short of the target (Algorithm 20 line 27: $\widehat{\gamma}_l > \gamma_l$ and $\widehat{\beta}_l > \beta_l$), the algorithm, if the current committee does not exceed a prefixed layer specific size limit, trains one more member to add to the committee.

Special attention should be paid here: the algorithm could alternate between case 2 and 3 and create a dead loop. A solution is to halve the unit u every time it happens until u becomes smaller than 10^{-5}. When it happens, one more round of training at this layer is recommended.

As mentioned earlier, to prevent a committee from growing too big, the algorithm stops refining its associated layer after a layer dependent size limit is breached (Algorithm 20 line 28). In this case, the shift $\mathfrak{s}$ is set to the smallest value that satisfies the false negative requirement. A harder learning case is thus deferred to the next layer. This strategy works because Adaboost's inability to meet the training target can often be explained by the fact that a classifier trained on a limited number of examples might not generalize well on the validation set. However, those hard negative patterns should ultimately appear and be learned if the training goes on, albeit one bit at a time.

To analyze how well the cascade does, let us assume that at layer l, Adaboost can deliver a classifier $f_{l,\mathfrak{s}_l}^{T_l}$ with false positive γ_l and detection rate $1-\beta_l$. In probabilistic terms, it means

$$\mathbb{P}(f_{l,\mathfrak{s}_l}^{T_l}(X)=1|Y=1) \geq 1-\beta_l \text{ and } \mathbb{P}(f_{l,\mathfrak{s}_l}^{T_l}(X)=1|Y=-1) \leq \gamma_l,$$

where by abuse of notation we keep $\mathbb{P}$ to denote the probability on some image space. If Algorithm 20 halts at the end of L iterations, the decision rule is

$$f_{cascade}(X) = 2\Big(\prod_{l=1}^{L} 1_{f_{l,\mathfrak{s}_l}^{T_l}(X)=1} - \frac{1}{2}\Big).$$

A window is thus declared positive if and only if all its component layers hold the same opinion

$$\begin{aligned}
&\mathbb{P}(f_{cascade}(X)=1|Y=-1)\\
=&\mathbb{P}\Big(\bigcap_{l=1}^{L}\{f_{l,\mathfrak{s}_l}^{T_l}(X)=1\}|Y=-1\Big)\\
=&\mathbb{P}(f_{L,\mathfrak{s}_L}^{T_L}(X)=1|\bigcap_{l=1}^{L-1}\{f_{l,\mathfrak{s}_l}^{T_l}(X)=1\} \text{ and } Y=-1)\mathbb{P}\Big(\bigcap_{l=1}^{L-1}\{f_{l,\mathfrak{s}_l}^{T_l}(X)=1\}|Y=-1\Big)\\
\leq&\gamma_l\mathbb{P}\Big(\bigcap_{l=1}^{L-1}\{f_{l,\mathfrak{s}_l}^{T_l}(X)=1\}|Y=-1\Big)\\
\leq&\gamma_l^L.
\end{aligned}$$

Likewise, the overall detection rate can be estimated as follows

$$
\begin{aligned}
&\mathbb{P}(f_{cascade}(X) = 1 | Y = 1) \\
=&\mathbb{P}(\bigcap_{l=1}^{L} \{f_{l,\mathfrak{s}_l}^{T_l}(X) = 1\} | Y = 1) \\
=&\mathbb{P}(f_{L,\mathfrak{s}_L}^{T_L}(X) = 1 | \bigcap_{l=1}^{L-1} \{f_{l,\mathfrak{s}_l}^{T_l}(X) = 1\} \text{ and } Y = 1) \mathbb{P}(\bigcap_{l=1}^{L-1} \{f_{l,\mathfrak{s}_l}^{T_l}(X) = 1\} | Y = 1) \\
\geq&(1 - \beta_l) \mathbb{P}(\bigcap_{l=1}^{L-1} \{f_{l,\mathfrak{s}_l}^{T_l}(X) = 1\} | Y = 1) \\
\geq&(1 - \beta_l)^L.
\end{aligned}
$$

In other words, if the empirically obtained rates are any indication, a faceless window will have a probability higher than $1 - \gamma_l$ to be labeled as such at each layer, which effectively directs the cascade classifier's attention on those more likely to have a face (see Figure 7.6).

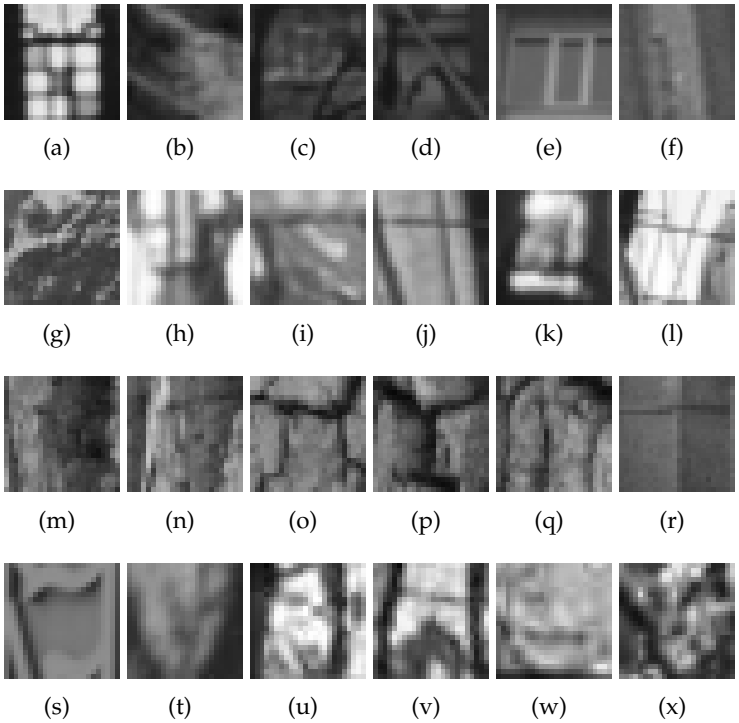
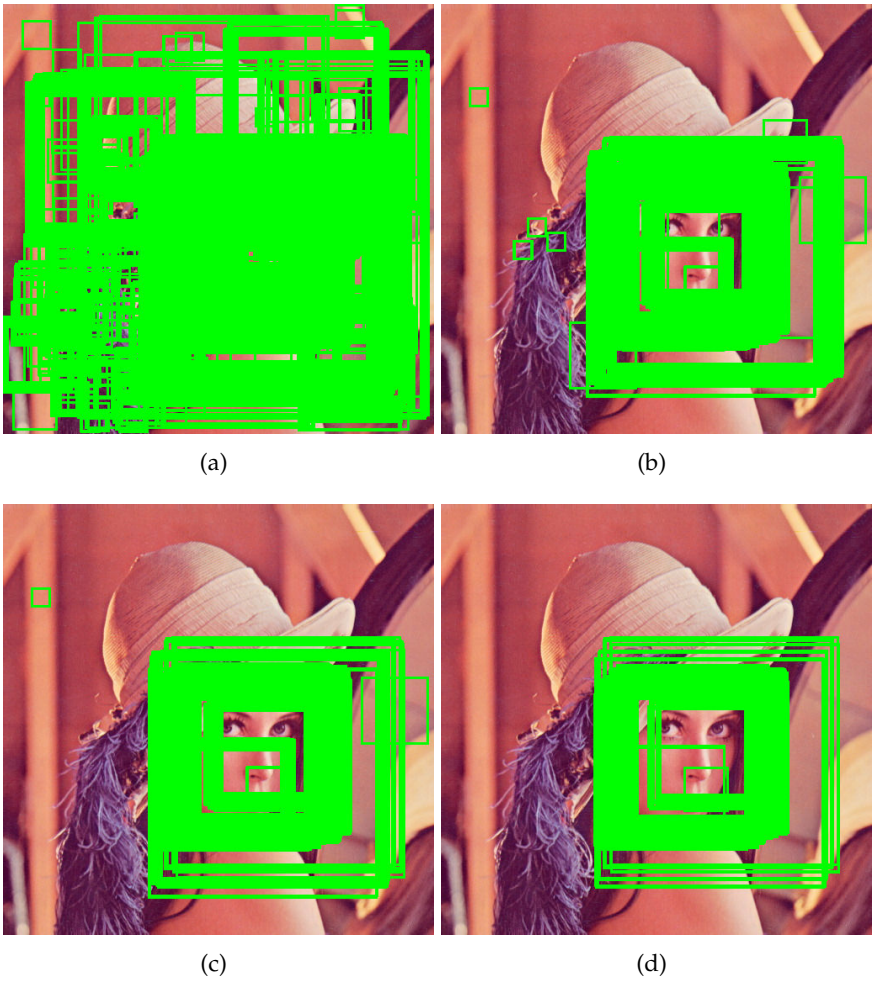


Figure 7.5 – *A selection of negative training examples at round 21 (a) (b) (c) (d) (e) (f), round 26 (g) (h) (i) (j) (k) (l), round 27 (m) (n) (o) (p) (q) (r), round 28 (s) (t) (u) (v) (w) (x). Observe how the negative training examples become increasingly difficult to discriminate from real faces.*

7.2.4 Dataset and experiments

A few words on the actual cascade training carried out on a 8-core Linux machine with 48G memory. We first downloaded 2897 different images without human faces from Jégou et al. (2008), Tkačik et al. (2011), Olmos (2004), National Oceanic and Atmospheric Administration (NOAA) Photo Library and European Southern Observatory. They were divided



(a) (b)

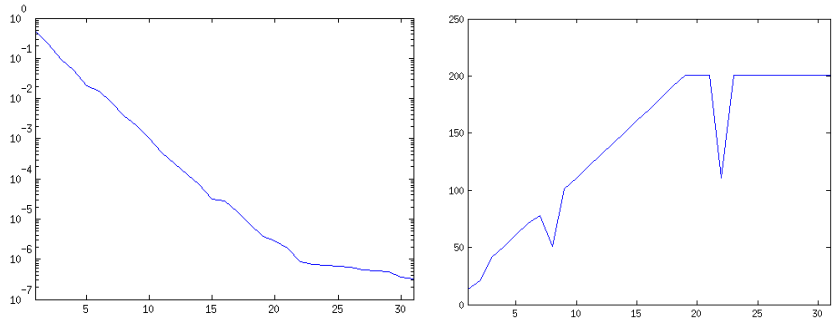
(c) (d)

Figure 7.6 – *How the trained cascade performs with (a) 16 layers, (b) 21 layers, (c) 26 layers and (d) 31 layers: the more layers, the less false positives.*

into two sets containing 2451 and 446 images respectively for training and validation. 1000 training and 1000 validation positive examples from an online source were used. The training process lasted for around 24 hours before producing a 31-layer cascade. It took this long because it became harder to get 2000 false positives (1000 for training and 1000 for validation) using Algorithm 19 with a more discriminative cascade: the algorithm needed to examine more images before it could come across enough good examples. The targeted false positive and false negative rate for each layer were set to 0.5 and 0.995 respectively and Figure 7.7 shows how the accumulated false positive rate as defined at line 12 and 33 of Algorithm 20 evolves together with the committee size. The fact that the later layers required more intensive training also contributed to a long training phase.

7.3 Post-processing

Figure 7.6(d) and Figure 7.8(a) show that the same face can be detected multiple times by a correctly trained cascade. This should come as no surprise as the positive examples (see Figure 7.3(a)) do allow a certain flexi-



(a) (b)

Figure 7.7 – *(a) Though occasionally stagnant, the accumulated false positive rate declines pretty fast with the number of layers. (b) As the learning task becomes more difficult as the cascade has more layers, more weak learners per layer are called upon. (In our experiments, the number of weak learners cannot exceed 201 per layer.)*

bility in pose and expression. On the contrary, many false positives do not enjoy this stability, despite the fact that taken out of context, some of them do look like a face (also see Figure 7.5). This observation lends support to the following *detection confidence* based heuristics for further reducing false positives and cleaning up the detected result (see Algorithm 21):

1. A detected window contains a face if and only if a sufficient number of other *adjacent* detected windows of the *same* size confirm it. To require windows of exactly the same size is not stringent because the test window sizes are quantified (see Algorithm 17 line 2). In this implementation, the window size multiplier is 1.5. Two $e \times e$ detected windows are said to be adjacent if and only if between the upper left corners of these two windows there is a path formed by the upper left corners of some detected windows of the same size. This condition is easily checked with the connected component algorithm [1] Shapiro and Stockman (2001). The number of test windows detecting a face is presumed to grow linearly with its size. This suggests the quotient of the cardinality of a connected component of adjacent windows by their common size e as an adequate confidence measure. This quotient is then compared to the scale invariant threshold empirically set at $3/24$, which means that to confirm the detection of a face of size 24×24, three adjacent detected windows are sufficient.
2. It is possible for the remaining detected windows to overlap after the previous test. In this case, we distinguish two scenarios (Algorithm 21 line 15–25):
   (a) If the smaller window's center is outside the bigger one, keep both.
   (b) Keep the one with higher detection confidence otherwise.

Finally, to make the detector slightly more rotation-invariant, in this implementation, we decided to run Algorithm 17 three times, once on

1. We obtained a version from `http://alumni.media.mit.edu/~rahimi/connected/`.

the input image, once on a clockwise rotated image and once on an anti-clockwise rotated image before post-processing all the detected windows (see Algorithm 22). In addition, when available, color also conveys valuable information to help further eliminate false positives. Hence in the current implementation, after the robustness test, an option is offered as to whether color images should be post-processed with this additional step (see Algorithm 23). If so, a detected window is declared positive only if it passes both tests.

Algorithm 21 Post-Processing

Input: a set G windows declared positive on an $M \times N$ grayscale image
Parameter: minimum detection confidence threshold $\mathfrak{r}$
Output: a reduced set of positive windows $\mathcal{P}$
Create an $M \times N$ matrix E filled with zeros.
for each window $w \in G$ **do**
    Take w's upper left corner coordinates (i, j) and its size e and set $E(i, j) \leftarrow e$
end for
Run a connected component algorithm on E.
for each component C formed by $|C|$ detected windows of dimension $e_C \times e_C$ **do**
    if its detection confidence $|C|e_C^{-1} > \mathfrak{r}$ **then**
        send one representing window to $\mathcal{P}$
    end if
end for
Sort the elements in $\mathcal{P}$ in ascending order of window size.
for window $i = 1$ to $|\mathcal{P}|$ **do**
    for window $j = i + 1$ to $|\mathcal{P}|$ **do**
        if window j remains in $\mathcal{P}$ and the center of window i is inside of window j **then**
            if window i has a higher detection confidence than window j **then**
                remove window j from $\mathcal{P}$
            else
                remove window i from $\mathcal{P}$ and break from the inner loop
            end if
        end if
    end for
end for
Return $\mathcal{P}$.

7.4 Appendix

This section explains, from a mathematical perspective, how and why Adaboost (Algorithm 16) works. We define the exponential loss

$$\forall (x, y) \in \mathbb{R}^d \times \{-1, 1\}, \quad L(y, f(x)) = \exp(-yf(x)),$$

where the classifier $f : \mathbb{R}^d \mapsto \mathbb{R}$ takes the form of a linear combination of weak classifiers

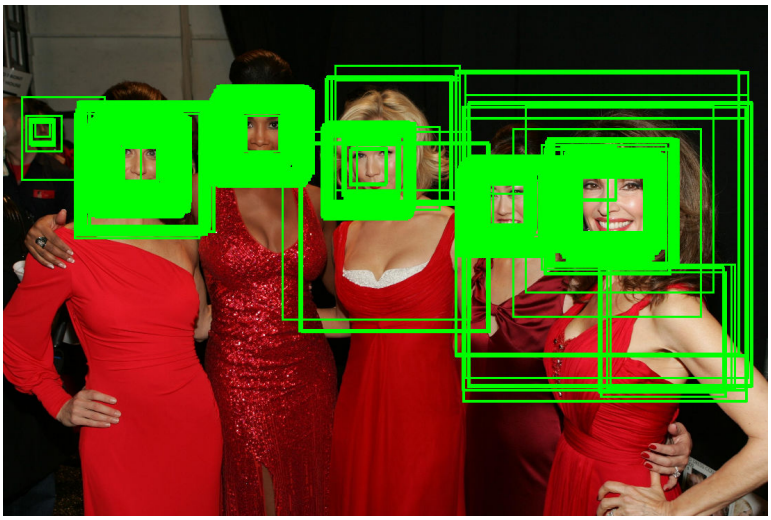
$$f(\cdot) = \sum_{t=1}^{T} \alpha_t h_t(\cdot),$$

Algorithm 22 Face detection with image rotation

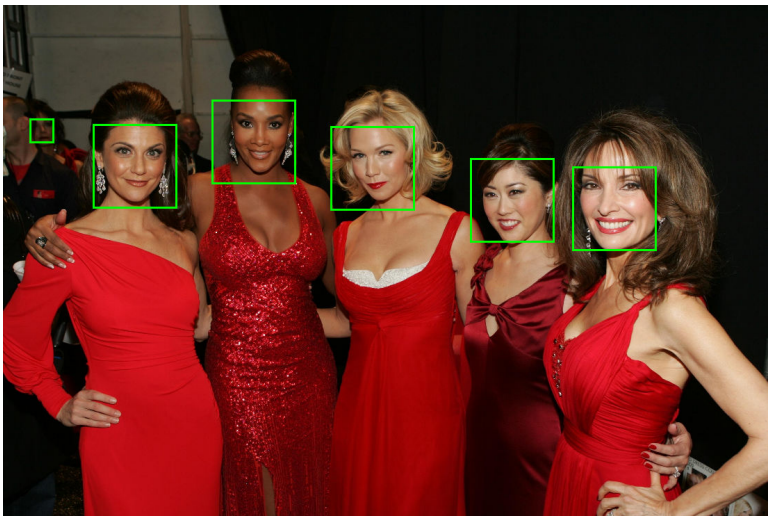
1: **Input:** an $M \times N$ grayscale image I
2: **Parameter:** rotation θ
3: **Output:** a set of detected windows $\mathcal{P}$
4: Rotate the image about its center by θ and $-\theta$ to have I_θ and $I_{-\theta}$
5: Run Algorithm 17 on I, I_θ and $I_{-\theta}$ to obtain three detected window sets $\mathcal{P}$, $\mathcal{P}_\theta$ and $\mathcal{P}_{-\theta}$ respectively
6: **for** each detected window w in $\mathcal{P}_\theta$ **do**
7: Get w's upper left corner's coordinates (i_w, j_w) and its size e_w
8: Rotate (i_w, j_w) about I_θ's center by $-\theta$ to get $(\tilde{i}_w, \tilde{j}_w)$
9: Quantify the new coordinates $\tilde{i}_w \leftarrow \min(\max(0, [\![\tilde{i}_w]\!]), M-1)$ and $\tilde{j}_w \leftarrow \min(\max(0, [\![\tilde{j}_w]\!]), N-1)$
10: **if** there is no $e_w \times e_w$ window located at $(\tilde{i}_w, \tilde{j}_w)$ in $\mathcal{P}$ **then**
11: Add it to $\mathcal{P}$
12: **end if**
13: **end for**
14: Replace θ by $-\theta$ and go through lines 6–13 again
15: Return $\mathcal{P}$

Algorithm 23 Skin Test

1: **Input:** an $N \times N$ color image I
2: **Output:** return whether I has enough skin like pixels
3: Set a counter $c \leftarrow 0$
4: **for** each pixel in I **do**
5: **if** the intensities of its green and blue channel are lower than that of its red channel **then**
6: $c \leftarrow c + 1$
7: **end if**
8: **end for**
9: **if** $c/N^2 > 0.4$ **then**
10: Return true
11: **else**
12: Return false
13: **end if**



(a)



(b)

Figure 7.8 – *The suggested post-processing procedure further eliminates a number of false positives and beautifies the detected result using a 31 layer cascade.*

with $T \in \mathbb{N}$, $\min_{1 \leq t \leq T} \alpha_t > 0$ and $\forall t,\ h_t(\cdot) \in \{-1, 1\}$. Naturally, the overall objective is

$$\min_{h_t, \alpha_t \geq 0} \sum_{i=1}^{n} w_i(1) L(y_i, \sum_{t=1}^{T} \alpha_t h_t(x_i)), \tag{7.5}$$

with some initial probabilistic weight $w_i(1)$. A greedy approach is deployed to deduce the optimal classifiers h_t and weights α_t one after another, although there is no guarantee that the objective (7.5) is minimized. Given $(\alpha_s, h_s)_{1 \leq s < t}$, let Z_{t+1} be the weighted exponential loss attained by a t-member committee and we seek to minimize it through (h_t, α_t)

$$
\begin{aligned}
Z_{t+1} &:= \min_{h_t,\alpha_t\geq 0} \sum_{i=1}^{n} w_i(1) e^{-y_i \sum_{s=1}^{t} \alpha_s h_s(x_i)} \\
&= \min_{h_t,\alpha_t\geq 0} \sum_{i=1}^{n} D_i(t) e^{-\alpha_t y_i h_t(x_i)} \\
&= \min_{h_t,\alpha_t\geq 0} \sum_{i=1}^{n} D_i(t) e^{-\alpha_t} 1_{y_i h_t(x_i)=1} + \sum_{i=1}^{n} D_i(t) e^{\alpha_t} 1_{y_i h_t(x_i)=-1} \\
&= \min_{h_t,\alpha_t\geq 0} e^{-\alpha_t} \sum_{i=1}^{n} D_i(t) + (e^{\alpha_t} - e^{-\alpha_t}) \sum_{i=1}^{n} D_i(t) 1_{y_i h_t(x_i)=-1} \\
&= Z_t \min_{h_t,\alpha_t\geq 0} e^{-\alpha_t} + (e^{\alpha_t} - e^{-\alpha_t}) \sum_{i=1}^{n} \frac{D_i(t)}{Z_t} 1_{y_i h_t(x_i)=-1},
\end{aligned}
$$

Therefore the optimization of Z_{t+1} can be carried out in two stages: first, because of α_t's assumed positivity, we minimize the weighted error using a base learning algorithm, a decision stump for instance

$$
\begin{aligned}
\epsilon_t &:= \min_h Z_t^{-1} \sum_{i=1}^{n} D_i(t) 1_{y_i h(x_i)=-1}, \\
h_t &:= \operatorname*{argmin}_h Z_t^{-1} \sum_{i=1}^{n} D_i(t) 1_{y_i h(x_i)=-1}.
\end{aligned}
$$

In case of multiple minimizers, take h_t to be any of them. Next choose

$$
\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t} = \operatorname*{argmin}_{\alpha>0} e^{-\alpha} + (e^{\alpha} - e^{-\alpha})\epsilon_t.
$$

Hence $\epsilon_t < 0.5$ is necessary, which imposes a minimal condition on the training set and the base learning algorithm. Also obtained is

$$
Z_{t+1} = 2Z_t \sqrt{\epsilon_t(1-\epsilon_t)} \leq Z_t,
$$

a recursive relation asserting the decreasing behavior of the exponential risk. Weight change thus depends on whether an observation is misclassified

$$
w_i(t+1) = \frac{D_i(t+1)}{Z_{t+1}} = \frac{D_i(t) e^{-y_i \alpha_t h_t(x_i)}}{2Z_t \sqrt{\epsilon_t(1-\epsilon_t)}} = \frac{w_i(t)}{2} \big(1_{h_t(x_i)=y_i} \frac{1}{1-\epsilon_t} + 1_{h_t(x_i)\neq y_i} \frac{1}{\epsilon_t} \big).
$$

The final boosted classifier is thus a weighted committee

$$
f(\cdot) := \operatorname{sign} \big[\sum_{t=1}^{T} \alpha_t h_t(\cdot) \big].
$$

Conclusions and perspectives

A patch based approach to image processing, like the binary classification problem examined in Chapter 7, can be framed as a risk minimization program. Examples of this formulation are abundant. Here let us focus on the denoising problem where a patch X and its noisy observation Y are related by

$$Y = X + N$$

with the Gaussian noise term N independent of X. The local performance of a filter f designed to estimate X from Y can be evaluated with the quadratic risk

$$R_X(f) := \mathbb{E}[\|X - f(Y)\|_2^2 | X].$$

Since X is rarely fixed in most applications, two overall risk criteria are used to define an optimal filter design. The first one is the minimax risk

$$\inf_f \sup_X R_X(f).$$

Given a basis in which X can be sparsely represented, this criterion leads to the remarkable shrinkage (thresholding) estimators Donoho and Johnstone (1994; 1995), which is then generalized to numerous variational formulations with ℓ_1 sparsity constraint Fornasier and Rauhut (2008) whose most rudimentary form is

$$f(Y) = D\alpha_* \text{ with } \alpha_* = \operatorname*{argmin}_\alpha \|Y - D\alpha\|_2^2 + \lambda\|\alpha\|_1$$

in which the dictionary D need not be an orthonormal basis and can be learned from data Elad and Aharon (2006), Mairal et al. (2009).

In contrast, this thesis adopts the minimal Bayesian risk criterion

$$\inf_f \mathbb{E} R_X(f),$$

which requires probabilistic modelling so as to make the meaning of the expectation $\mathbb{E}$ precise. Note that a Bayesian optimal filter is not necessarily minimax and vice versa. But it has an explicit probabilistic interpretation because of the quadratic risk:

$$\operatorname*{argmin}_f \mathbb{E} R_X(f) = \operatorname*{argmin}_f \mathbb{E}\|X - f(Y)\|_2^2 = \mathbb{E}[X|Y].$$

We now review several parametric approximations of this Bayesian optimal filter (see Chapter 5 for the non-parametric ones). Note again that Bayesian optimality hinges on X's distribution.

1. as $\mathbb{E}[X|Y]$ only depends on the posterior distribution $\mathbb{P}(X|Y)$, if we model it as a multivariate Gaussian and infer its parameters from similar patches in the same image, it results in the highly successful NL-Bayes Lebrun et al. (2013b).
2. an indirect approach to get $\mathbb{P}(X|Y)$ is to model the marginal law of X, or the Bayesian prior. Ideally this prior should be the patch composition of the image to restore $(X_i)_{1\leq i\leq N}$ because then the Bayesian risk $\mathbb{E}R_X(f)$ equals the expected patch MSE

$$\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[\|X_i - f(Y_i)\|_2^2|X_i] \tag{7.6}$$

   which, as argued in Chapter 1, serves as an asymptotic upper bound of the expected image MSE.

   This motivates a Gaussian mixture modelling because of the ease with which such a device can fit patches extracted locally from noisy images thanks to the Expectation and Maximization algorithm. The use of Gaussian mixture also produces a closed form of the conditional expectation as a weighted sum of Wiener filters (see Chapter 1, 5). However, the fact that such a filter did not yield superior numerical results indicates the limitation of the modelling premise. Nonetheless, it enables a posterior likelihood based patch classification and results in an over-complete dictionary for sparse patch representation. In addition, thanks to SURE, the Bayesian risk (7.6) can be estimated, which aids filter selection within the Bayesian risk minimization paradigm.

   Without SURE guided filter selection, the main conceptual difference between this algorithm and NL-Bayes is that the former is less conservative in its inference of $\mathbb{P}(X|Y)$ as it involves patches with similar orientations rather than just similar patches. Their performance gap thus highlights the fundamental difficulty of image denoising due to the rather varying nature of $Y \mapsto \mathbb{P}(X|Y)$: it is not smooth from one patch to another within the same orientation class.
3. to circumvent the difficulty of inferring from a single noisy image its prior patch distribution, we may use a more complex patch distribution from a generic image database Levin and Nadler (2011), Burger et al. (2012). In so doing, we set the stage for the definitive Bayesian framework as a filter's Bayesian risk now represents its aggregate performance on a wide-range of natural images. Clearly, the benefit brought by this approach is unlimited noise-free data. With multilayer neural networks, it seems finally possible to get our hands on this Bayesian optimal filter. But the Bayesian nature also has its own share of issues. For instance, it is reported Burger et al. (2012) that multilayer neural networks have trouble restoring regular but high frequency patterns, which is unsurprising given their relative scarcity (see Chapter 1). Hence the ability to evaluate a pat-

tern's Bayesian prior is desirable as it allows to switch between the Bayesian and minimax filters for a better result.

Finally, we note that larger observations help reduce the Bayesian risk, as shown by the conditional expectation argument (see Chapter 5). But for all practical purposes, we should overcome the temptation to employ ever larger neural networks. Chapter 4, 5 are such examples which make use of the image patch distribution's special properties. Furthermore, an analogy between the conditional expectation and an ordinary function indicates two other possibilities:

1. as in linear interpolation, we may partition a complex neural network's domain of definition so as to approximate it with several simpler ones with restricted domains;
2. as in function approximation, we may consider some hand crafted algorithms as a first approximation and use neural networks exclusively for computing the difference.

Bibliography

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: Design of dictionaries for sparse representation. *Proceedings of Signal Processing with Adaptive Sparse Structured Representations*, 5:9–12, 2005. http://dx.doi.org/10.1109/TSP.2006.881199.

A. R. Barron. Approximation and estimation bounds for artificial neural networks. *Machine Learning*, 14(1):115–133, 1994. http://dx.doi.org/10.1007/BF00993164.

B. E. Bayer. Color imaging array, Juillet 20 1976. US Patent 3,971,065.

Y. Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009. http://dx.doi.org/10.1561/2200000006.

M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 417–424. ACM Press, 2000. http://dx.doi.org/10.1145/344779.344972.

M. Bertalmio, L. Vese, G. Sapiro, and S. Osher. Simultaneous structure and texture image inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003. http://dx.doi.org/10.1109/TIP.2003.815261.

L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Symposium on Computational Statistics*, pages 177–186. Springer, 2010. http://dx.doi.org/10.1007/978-3-7908-2604-3_16.

P. J. Brockwell and R. A. Davis. Time Series: Theory and Models, 1991.

A. Buades, B. Coll, and J. M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005. http://dx.doi.org/10.1145/344779.344972.

A. Buades, B. Coll, and J. M. Morel. Non-Local Means Denoising. *Image Processing On Line*, 2011a. http://dx.doi.org/10.5201/ipol.2011.bcm_nlm.

A. Buades, B. Coll, J. M. Morel, and C. Sbert. Self-similarity driven color demosaicking. *IEEE Transactions on Image Processing*, 18(6):1192–1202, 2009. http://dx.doi.org/10.1109/TIP.2009.2017171.

A. Buades, B. Coll, J. M. Morel, and C. Sbert. Self-similarity Driven Demosaicking. *Image Processing On Line*, 2011b.

H. C. Burger, C. Schuler, and S. Harmeling. Learning how to combine internal and external denoising methods. In *Pattern Recognition*, pages 121–130. Springer, 2013.

H. C. Burger, C. J. Schuler, and S. Harmeling. Image denoising: Can plain neural networks compete with BM3D? In *Computer Vision and Pattern Recognition*, pages 2392–2399. IEEE, 2012. http://dx.doi.org/10.1109/CVPR.2012.6247952.

H.C. Burger. *Modelling and Learning Approaches to Image Denoising*. PhD thesis, Eberhard Karls Universität Tübingen, Wilhelmstr. 32, 72074 Tübingen, 2013.

Emmanuel J Candès. What is... a curvelet? *Notices of the American Mathematical Society*, 50(11):1402–1403, 2003.

P. Chatterjee, N. Joshi, S. B. Kang, and Y. Matsushita. Noise suppression in low-light images through joint denoising and demosaicing. In *Computer Vision and Pattern Recognition*, pages 321–328. IEEE, 2011. http://dx.doi.org/10.1109/CVPR.2011.5995371.

P. Chatterjee and P. Milanfar. Clustering-based denoising with locally learned dictionaries. *IEEE Transactions on Image Processing*, 18(7):1438–1451, 2009. http://dx.doi.org/10.1109/TIP.2009.2018575.

S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, 20(1):33–61, 1998. http://dx.doi.org/10.1137/S1064827596304010.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007. http://dx.doi.org/10.1109/TIP.2007.901238.

I. Daubechies et al. *Ten lectures on wavelets*, volume 61. SIAM, 1992.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*, volume 31. Springer, 1996. http://dx.doi.org/10.1007/978-1-4612-0711-5.

D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994. http://dx.doi.org/10.2307/2337118.

D. L. Donoho and I. M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995. http://dx.doi.org/10.2307/2291512.

B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, volume 2, pages 1033–1038. IEEE, 1999. http://dx.doi.org/10.1109/ICCV.1999.790383.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006. http://dx.doi.org/10.1109/TIP.2006.881969.

M. Elad, J.L. Starck, P. Querre, and D.L. Donoho. Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA). *Applied and Computational Harmonic Analysis*, 19(3):340–358, 2005. http://dx.doi.org/10.1016/j.acha.2005.03.005.

M. Fornasier and H. Rauhut. Iterative thresholding algorithms. *Applied and Computational Harmonic Analysis*, 25(2):187–208, 2008. http://dx.doi.org/10.1016/j.acha.2007.10.005.

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. http://dx.doi.org/10.1006/jcss.1997.1504.

Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal of Japanese Society For Artificial Intelligence*, 14:771–780, 1999.

J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer Series in Statistics, 2001.

P. Getreuer. Contour stencils: Total variation along curves for adaptive image interpolation. *SIAM Journal on Imaging Sciences*, 4(3):954–979, 2011a. http://dx.doi.org/10.1137/100802785.

P. Getreuer. Zhang-Wu Directional LMMSE Image Demosaicking. *Image Processing On Line*, 2011b.

P. Getreuer. Image Demosaicking with Contour Stencils. *Image Processing On Line*, 2012.

J. Go, K. Sohn, and C. Lee. Interpolation using neural networks for digital still cameras. *IEEE Transactions on Consumer Electronics*, 46(3):610–616, 2000. http://dx.doi.org/10.1109/30.883419.

J. F. Hamilton and J. E. Adams. Adaptive color plan interpolation in single sensor color electronic camera, Mai 13 1997. US Patent 5,629,734.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, page 50. Manchester, UK, 1988. http://dx.doi.org/10.5244/C.2.23.

K. Hirakawa and T. W. Parks. Joint demosaicing and denoising. *IEEE Transactions on Image Processing*, 15(8):2146–2157, 2006. http://dx.doi.org/10.1109/TIP.2006.875241.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. http://dx.doi.org/10.1016/0893-6080(89)90020-8.

V. Jain and S. Seung. Natural image denoising with convolutional networks. In *Advances in Neural Information Processing Systems*, pages 769–776, 2009.

H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*, volume I from *Lecture Notes in Computer Science*, pages 304–317. Springer, 2008. http://dx.doi.org/10.1007/978-3-540-88682-2_24.

O. Kapah and H. Z. Hel-Or. Demosaicking using artificial neural networks. In *Electronic Imaging*, pages 112–120. International Society for Optics and Photonics, 2000. http://dx.doi.org/10.1117/12.382904.

S. M. Kay. Fundamentals of statistical signal processing: estimation theory. 1993.

M. Lebrun. An Analysis and Implementation of the BM3D Image Denoising Method. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.l-bm3d.

M. Lebrun, A. Buades, and J. M. Morel. Implementation of the "Non-Local Bayes" (NL-Bayes) Image Denoising Algorithm. *Image Processing On Line*, 2013a. http://dx.doi.org/10.5201/ipol.2013.16.

M. Lebrun, A. Buades, and J. M. Morel. A nonlocal bayesian image denoising algorithm. *SIAM Journal on Imaging Sciences*, 6(3):1665–1688, 2013b. http://dx.doi.org/10.1137/120874989.

M. Lebrun and A. Leclaire. An Implementation and Detailed Analysis of the K-SVD Image Denoising Algorithm. *Image Processing On Line*, 2012. http://dx.doi.org/10.5201/ipol.2012.llm-ksvd.

Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–50. Springer, 1998. http://dx.doi.org/10.1007/3-540-49430-8_2.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

A. Levin and B. Nadler. Natural image denoising: Optimality and inherent bounds. In *Computer Vision and Pattern Recognition*, pages 2833–2840. IEEE, 2011. http://dx.doi.org/10.1109/CVPR.2011.5995309.

X. Li, B. Gunturk, and L. Zhang. Image demosaicking: a systematic survey. In *Visual Communications and Image Processing*, volume 6822, pages 68221J–68221J, 2008. http://dx.doi.org/10.1117/12.766768.

J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012. http://dx.doi.org/10.1109/TPAMI.2011.156.

J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *International Conference on Computer Vision*, pages 2272–2279. IEEE, 2009. http://dx.doi.org/10.1109/ICCV.2009.5459452.

S. Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993. http://dx.doi.org/10.1109/78.258082.

S. Masnou and J. M. Morel. Level lines based disocclusion. In *International Conference on Image Processing*, pages 259–263. IEEE, 1998. http://dx.doi.org/10.1109/ICIP.1998.999016.

J. M. Morel and G. Yu. Is SIFT scale invariant? *Inverse Problems and Imaging*, 5(1):115–136, 2011. http://dx.doi.org/10.3934/ipi.2011.5.115.

J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980. http://dx.doi.org/10.2307/2006193.

A. Olmos. A biologically inspired algorithm for the recovery of shading and reflectance images. *Perception*, 33(12):1463–1473, 2004. http://dx.doi.org/10.1068/p5321.

B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381 (6583):607–609, 1996. http://dx.doi.org/10.1038/381607a0.

D. Paliy, V. Katkovnik, R. Bilcu, S. Alenius, and K. Egiazarian. Spatially adaptive color filter array interpolation for noiseless and noisy data. *International Journal of Imaging Systems and Technology*, 17(3):105–122, 2007. http://dx.doi.org/10.1002/ima.20109.

P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990. http://dx.doi.org/10.1109/34.56205.

J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000. http://dx.doi.org/10.1023/A:1026553619983.

J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli. Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2003. http://dx.doi.org/10.1109/TIP.2003.818640.

B. Rajaei. An Analysis and Improvement of the BLS-GSM Denoising Method. *Image Processing On Line*, 2014. http://dx.doi.org/10.5201/ipol.2014.86.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958. http://dx.doi.org/10.1037/h0042519.

S. Roweis. EM algorithms for PCA and SPCA. *Advances in Neural Information Processing Systems*, pages 626–632, 1998.

L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992. http://dx.doi.org/10.1016/0167-2789(92)90242-F.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998. http://dx.doi.org/10.1214/aos/1024691352.

L. Shapiro and G. C. Stockman. Computer Vision, 2001.

J. Shen and T. F. Chan. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002. http://dx.doi.org/10.1137/S0036139900368844.

E. P. Simoncelli and W. T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *International Conference on Image Processing*, volume 3, pages 3444–3444. IEEE Computer Society, 1995. http://dx.doi.org/10.1109/ICIP.1995.537667.

J.L. Starck, E. J Candès, and D. L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):670–684, 2002. http://dx.doi.org/10.1109/TIP.2002.1014998.

C. M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, pages 1135–1151, 1981. http://dx.doi.org/10.1214/aos/1176345632.

H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing*, 16(2):349–366, 2007. http://dx.doi.org/10.1109/TIP.2006.888330.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):443–482, 1999a. http://dx.doi.org/10.1162/089976699300016728.

M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999b. http://dx.doi.org/10.1111/1467-9868.00196.

G. Tkačik, P. Garrigan, C. Ratliff, G. Milčinski, J. M. Klein, L. H. Seyfarth, P. Sterling, D. H. Brainard, and V. Balasubramanian. Natural images from the birthplace of the human eye. *Public Library of Science One*, 6(6):e20409, 2011.

C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision*, pages 839–846. IEEE, 1998. http://dx.doi.org/10.1109/ICCV.1998.710815.

P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008. http://dx.doi.org/10.1145/1390156.1390294.

P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004. http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008. http://dx.doi.org/10.1561/2200000001.

Y. Q. Wang. E-PLE: an Algorithm for Image Inpainting. *Image Processing On Line*, 2013a. http://dx.doi.org/10.5201/ipol.2013.54.

Y. Q. Wang. The Implementation of SURE Guided Piecewise Linear Image Denoising. *Image Processing On Line*, 2013b. http://dx.doi.org/10.5201/ipol.2013.52.

J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Advances in Neural Information Processing Systems*, pages 341–349, 2012.

G. Yu and G. Sapiro. DCT image denoising: a simple and effective image denoising algorithm. *Image Processing On Line*, 2011. http://dx.doi.org/10.5201/ipol.2011.ys-dct.

G. Yu, G. Sapiro, and S. Mallat. Solving inverse problems with piecewise linear estimators: from Gaussian mixture models to structured sparsity. *IEEE Transactions on Image Processing*, 21(5):2481–2499, 2012. http://dx.doi.org/10.1109/TIP.2011.2176743.

H. Yue, X. Sun, J. Yang, and F. Wu. CID: Combined Image Denoising in Spatial and Frequency Domains Using Web Images. In *Computer Vision and Pattern Recognition*, pages 2933–2940. IEEE, 2014. http://dx.doi.org/10.1109/CVPR.2014.375.

L. Zhang and X. Wu. Color demosaicking via directional linear minimum mean square-error estimation. *IEEE Transactions on Image Processing*, 14(12):2167–2178, 2005. http://dx.doi.org/10.1109/TIP.2005.857260.

L. Zhang, X. Wu, A. Buades, and X. Li. Color demosaicking by local directional interpolation and nonlocal adaptive thresholding. *Journal of Electronic Imaging*, 20(2):023016–023016, 2011.

D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision*, pages 479–486. IEEE, 2011. http://dx.doi.org/10.1109/ICCV.2011.6126278.