

Designing Partial Differential Equations for Image Processing by Combining Differential Invariants*

Zhouchen Lin^{1†} Wei Zhang² Xiaoou Tang²

¹Microsoft Research Asia, zhoulin@microsoft.com

²The Chinese University of Hong Kong, {zw007,xtang}@ie.cuhk.edu.hk

Abstract

Partial differential equations (PDEs) have been successful for solving many problems in image processing and computer vision. However, designing PDEs usually requires high mathematical skills and good insight to the problems. In this paper, we propose a framework for learning a system of PDEs from real data. Compared to the traditional approaches to designing PDEs, our framework requires much less human intelligence. We assume that the system consists of two PDEs. One controls the evolution of the output and the other is for an *indicator function* that helps collect global information. As the PDEs should be shift and rotationally invariant, they must be functions of differential invariants that are shift and rotationally invariant. Currently, we assume that the PDEs are simply linear combinations of the fundamental differential invariants up to second order. The combination coefficients can be learnt from real data via an optimal control technique. The exemplary experiments show that the PDEs, designed in our unified way, can solve some image processing problems reasonably well. Hence our framework is very promising. We expect that with future improvements our framework could work for more image processing/computer vision problems.

1 Introduction

The applications of partial differential equations (PDEs) to computer vision and image processing date back to the 1960s [9, 12]. However, this technique did not draw much attention until the introduction of the concept of scale space by Koenderink [14] and Witkin [29] in the 1980s. Perona and Malik's work on anisotropic diffusion [23] further drew great interest from researchers towards PDE-based methods. Nowadays, PDEs have been successfully applied to many problems in image processing and computer vision [11, 8, 25, 2, 7], e.g.,

*Microsoft technical report #MSR-TR-2009-192. Patent filed.

†Corresponding author.

denoising [23], enhancement [20], inpainting [4], segmentation [15], stereo and optical flow computation.

In general, there are three kinds of methods used to design PDEs. For the first kind of methods, PDEs are written down directly, based on some mathematical understandings on the properties of the PDEs (e.g., anisotropic diffusion [23], shock filter [20] and curve evolution based equations [11, 25, 2, 7]). The second kind of methods basically define an energy functional first, which collects the wish list of the desired properties of the output image, and then derives the evolution equations by computing the Euler-Lagrange equation of the energy functional (e.g., chapter 9 of [11]). The third kind of methods are axiomatic approaches, which first prescribe the properties, i.e., axioms, that the PDEs should hold and then deduce the form of PDEs from these axioms (e.g., Alvarez et al.’s axiomatic formulation of the scale-space theory [1]). All of these methods require both good insight to what properties to hold and high mathematical skills, in order to acquire the desired PDEs. These greatly limit the applications of PDEs to wider and more complex scopes. This motivates us to explore whether there is an easier way of designing PDEs.

In this paper, we show that learning PDEs from given input-output sample image pairs might be a possible way to designing PDEs in a lazy manner. The key idea is to assume that the PDEs in sought could be written as combinations of “atoms” called fundamental differential invariants. Then the problem boils down to determining the combination coefficients among such “atoms”. This can be achieved by employing the technique of optimal control governed by PDEs [17], where the objective functional is to minimize the difference between the expected outputs (ground truth) and the actual outputs of the PDEs, given the input images. Such input-output image pairs are provided by the user as training samples. As a preliminary investigation, we assume that the system consists of two PDEs. One controls the evolution of the output and the other is for an *indicator function* that helps collect global information. As the PDEs should be shift and rotationally invariant, they must be functions of fundamental differential invariants [19]. Currently, we only consider the case that the PDEs are linear combinations of fundamental differential invariants up to second order, and

the objective functional simply utilizes the L_2 norm to measure the output error.

Differential invariants are already widely used in computer vision and image processing. However, the existing use of differential invariants is mainly limited to feature representation, detection and matching [18]. Anisotropic PDE-based filtering of images [7, 8, 25] also uses differential invariants. However, those PDEs are for image denoising and enhancement only, and the deduction of affine and perspective differential invariants therein is rather involved. Although our differential invariants are much simpler, we can obtain PDEs that are applicable to more image processing problems.

The optimal control technique has also been applied to some computer vision and image processing problems. In [13], Kimia et al. showed how to use this technique to solve energy functionals arising from various problems, including shape evolution, morphology, optical flow estimation and shape from shading, whose variables are governed by (partial) differential equations. Papadakis et al. also applied the optimal control technique extensively (e.g., optical flow estimation [22] and tracking [21]). Our framework is different from the existing work in that it is to determine the (coefficients in) PDEs, while the existing work is to determine PDE *outputs*, in which the PDE is known.

We have to emphasize that we do *not* claim that our current framework, in its preliminary form, can produce PDEs for *all* image processing/computer vision problems. However, we show that it does work for some image processing problems: it produces PDEs that perform reasonably well for these problems. The merit of our framework is that the PDEs are acquired using the same mechanism. The only differences are in the sets of training images that the user has to prepare and the computed combination coefficients among the fundamental differential invariants. We do find problems that the current framework fails. Nonetheless, the successful examples encourage us to further improve our framework in the future so that it can handle more problems systematically.

The rest of this paper is organized as follows. In Section 2, we present our framework of learning PDEs. In Section 3, we testify to the effectiveness of our framework by applying it to five basic image processing problems. Then we conclude our paper in Section 4.

Table 1: Notations

\mathbf{x}	(x, y) , spatial variable	t	temporal variable
Ω	an open region of R^2	$\partial\Omega$	boundary of Ω
Q	$\Omega \times (0, T)$	Γ	$\partial\Omega \times (0, T)$
∇f	gradient of f	\mathbf{H}_f	Hessian of f
$\langle f \rangle$	$\{f, f_x, f_y, f_{xx}, f_{xy}, f_{yy}, \dots\}$, i.e., the set of partial derivatives of f up to an appropriate order, where $f_x = \partial f / \partial x$, etc.		

2 Designing PDEs by Combining Differential Invariants

Now we present our framework of learning a PDE system from training sample images. We assume that the PDE system consists of two PDEs. One is for the evolution of the output image O , and the other is for the evolution of an *indicator function* ρ . The goal of introducing the indicator function is to collect large scale information in the image so that the evolution of O can be correctly guided. Although the introduction of the indicator function is inspired by the edge indicator in [11] (page 193), it may not necessarily be related to the edge information for non-edge-detection problems. We assume the PDEs to be of evolutionary type because a usual information process should consist of some (unknown) steps. The time-dependent operations of the evolutionary PDEs resemble the different steps of the information process. Moreover, for stationary PDEs it is not natural to define their inputs and outputs, and the existence of their solutions is much less optimistic. So our PDE system can be written as:

$$\left\{ \begin{array}{ll} \frac{\partial O}{\partial t} = L_O(\mathbf{a}, \langle O \rangle, \langle \rho \rangle), & (\mathbf{x}, t) \in Q, \\ O = 0, & (\mathbf{x}, t) \in \Gamma, \\ O|_{t=0} = O_0, & \mathbf{x} \in \Omega; \\ \frac{\partial \rho}{\partial t} = L_\rho(\mathbf{b}, \langle \rho \rangle, \langle O \rangle), & (\mathbf{x}, t) \in Q, \\ \rho = 0, & (\mathbf{x}, t) \in \Gamma, \\ \rho|_{t=0} = \rho_0, & \mathbf{x} \in \Omega. \end{array} \right. \quad (1)$$

The notations in the above equations can be found in Table 1, where Ω is the rectangular region occupied by the input image I and T is the time that the PDE system finishes the processing and outputs the results. For computational issues and ease of mathematical deduction, I will be padded with zeros of several pixels width around it. As we can change the

Table 2: Shift and rotationally invariant fundamental differential invariants up to second order.

i	$\text{inv}_i(\rho, O)$
0,1,2	$1, \rho, O$
3	$\ \nabla\rho\ ^2 = \rho_x^2 + \rho_y^2$
4	$(\nabla\rho)^t \nabla O = \rho_x O_x + \rho_y O_y$
5	$\ \nabla O\ ^2 = O_x^2 + O_y^2$
6,7	$\text{tr}(\mathbf{H}_\rho) = \rho_{xx} + \rho_{yy}, \text{tr}(\mathbf{H}_O) = O_{xx} + O_{yy}$
8	$(\nabla\rho)^t \mathbf{H}_\rho \nabla\rho = \rho_x^2 \rho_{xx}^2 + 2\rho_x \rho_y \rho_{xy}^2 + \rho_y^2 \rho_{yy}^2$
9	$(\nabla\rho)^t \mathbf{H}_O \nabla\rho = \rho_x^2 O_{xx}^2 + 2\rho_x \rho_y O_{xy}^2 + \rho_y^2 O_{yy}^2$
10	$(\nabla\rho)^t \mathbf{H}_\rho \nabla O = \rho_x O_x \rho_{xx} + (\rho_y O_x + \rho_x O_y) \rho_{xy} + \rho_y O_y \rho_{yy}$
11	$(\nabla\rho)^t \mathbf{H}_O \nabla O = \rho_x O_x O_{xx} + (\rho_y O_x + \rho_x O_y) O_{xy} + \rho_y O_y O_{yy}$
12	$(\nabla O)^t \mathbf{H}_\rho \nabla O = O_x^2 \rho_{xx} + 2O_x O_y \rho_{xy} + O_y^2 \rho_{yy}$
13	$(\nabla O)^t \mathbf{H}_O \nabla O = O_x^2 O_{xx} + 2O_x O_y O_{xy} + O_y^2 O_{yy}$
14	$\text{tr}(\mathbf{H}_\rho^2) = \rho_{xx}^2 + 2\rho_{xy}^2 + \rho_{yy}^2$
15	$\text{tr}(\mathbf{H}_\rho \mathbf{H}_O) = \rho_{xx} O_{xx} + 2\rho_{xy} O_{xy} + \rho_{yy} O_{yy}$
16	$\text{tr}(\mathbf{H}_O^2) = O_{xx}^2 + 2O_{xy}^2 + O_{yy}^2$

unit of time, it is harmless to fix $T = 1$. L_O and L_ρ are smooth functions. O_0 and ρ_0 are the initial functions of O and ρ , respectively. $\mathbf{a} = \{a_i\}$ and $\mathbf{b} = \{b_i\}$ are sets of functions defined on Q that are used to control the evolution of O and ρ , respectively. The forms of L_O and L_ρ are discussed below.

2.1 Forms of PDEs

The space of all PDEs is infinite dimensional. To find the right form, we start with the properties that our PDE system should have, in order to narrow down the search space. We notice that most image processing problems are shift and rotationally invariant, i.e., when the input image is shifted or rotated, the output image is also shifted or rotated by the same amount. So we require that our PDE system is shift and rotationally invariant. In the work of Alvarez et al. [1], different forms of PDEs are determined by assuming various transformational invariance. In our framework, as we want to make our method applicable to more problems, we only assume the shift and rotational invariance. So unlike [1], our form of PDEs is not unique.

Then according to the differential invariant theory in [19], L_O and L_ρ must be functions

of the fundamental differential invariants under the groups of shift and rotation. The fundamental differential invariants are invariant under shift and rotation and other invariants can be written as their functions. The set of fundamental differential invariants is not unique, but different sets can express each other. We should choose invariants in the simplest form in order to ease mathematical deduction and analysis and numerical computation. Fortunately, for shift and rotational invariance, the fundamental differential invariants can be chosen as polynomials of the partial derivatives. We list those up to second order in Table 2.¹ As ∇f and \mathbf{H}_f change to $\mathbf{R}\nabla f$ and $\mathbf{R}\mathbf{H}_f\mathbf{R}^t$, respectively, when the image is rotated by a matrix \mathbf{R} , it is easy to check the rotational invariance of the quantities in Table 2. In the sequel, we shall use $\text{inv}_i(\rho, O)$, $i = 0, 1, \dots, 16$, to refer to them in order. Note that those invariants are ordered with ρ going before O . We may reorder them with O going before ρ . In this case, the i -th invariant will be referred to as $\text{inv}_i(O, \rho)$.

Moreover, for L_O and L_ρ to be shift invariant, the control functions a_i and b_i must be independent of \mathbf{x} , i.e., they must be functions of t only. The proof is presented in Appendix 1.

So the simplest choice of functions L_O and L_ρ is the linear combination of these differential invariants, leading to the following forms:

$$\begin{aligned} L_O(\mathbf{a}, \langle O \rangle, \langle \rho \rangle) &= \sum_{j=0}^{16} a_j(t) \text{inv}_j(\rho, O), \\ L_\rho(\mathbf{b}, \langle \rho \rangle, \langle O \rangle) &= \sum_{j=0}^{16} b_j(t) \text{inv}_j(O, \rho). \end{aligned} \tag{2}$$

Currently we only limit our attention to second order PDEs because most of the PDE theories are of second order and most PDEs arising from engineering are also of second order. It will pose difficulties in theoretical analysis and numerical computation if higher order PDEs are considered. So we choose to leave the involvement of higher order derivatives to future work.

Note that we do *NOT* claim that the real information process obeys our PDEs. We only care whether the *final output* of our PDE system, i.e., $O(\mathbf{x}, 1)$, can *approximate* that of

¹We add the constant function “1” for the convenience of mathematical deduction in the sequel.

the real process. For example, although $O_1(\mathbf{x}, t) = \|\mathbf{x}\|^2 \sin t$ and $O_2(\mathbf{x}, t) = (\|\mathbf{x}\|^2 + (1 - t)\|\mathbf{x}\|)(\sin t + t(1 - t)\|\mathbf{x}\|^3)$ are very different functions, they initiate from the same function at $t = 0$ and also settle down at the same function at time $t = 1$. So both functions fit our purpose and we need not care whether the real process obeys either function.

2.2 Determining the Coefficients by the Optimal Control Approach

Given the forms of PDEs shown in (2), we have to determine the coefficient functions $a_j(t)$ and $b_j(t)$. We may prepare training samples (I_m, \tilde{O}_m) , where I_m is the input image and \tilde{O}_m is the expected output image, $m = 1, 2, \dots, M$, and compute the coefficient functions that minimize the following functional:

$$\begin{aligned} & J\left(\{O_m\}_{m=1}^M, \{a_j\}_{j=0}^{16}, \{b_j\}_{j=0}^{16}\right) \\ &= \frac{1}{2} \sum_{m=1}^M \int_{\Omega} [O_m(\mathbf{x}, 1) - \tilde{O}_m(\mathbf{x})]^2 d\Omega + \frac{1}{2} \sum_{j=0}^{16} \lambda_j \int_0^1 a_j^2(t) dt + \frac{1}{2} \sum_{j=0}^{16} \mu_j \int_0^1 b_j^2(t) dt, \end{aligned} \quad (3)$$

where $O_m(\mathbf{x}, 1)$ is the output image at time $t = 1$ computed from (1) when the input image is I_m , and λ_j and μ_j are positive weighting parameters. The first term of (3) requires that the final output of our PDE system be close to the ground truth. The second and the third terms of (3) are for regularization, so that the optimal control problem is well posed. The regularization is important in inverse problems. It is particularly useful for our problem as the number of training samples may be limited.

Then the (locally) optimal $\{a_j\}$ and $\{b_j\}$ can be computed by speediest descent: at each iteration the Gâteaux derivatives of J with respect to a_j and b_j can be evaluated with the help of the adjoint equations of (1)-(2) (see Appendix 2), then $\{a_j\}$ and $\{b_j\}$ are updated after determining an optimal stepsize along the “negative direction” of the Gâteaux derivatives. The reader may refer to [17] for a rigorous exposition on optimal control theory governed by PDEs, or refer to [16] and other sources, e.g., [21, 22], for a friendly introduction.

2.3 Implementation Details

2.3.1 Discretization

We solve (1)-(2) by finite difference. The spatial stepsizes Δx and Δy are both normalized to 1 (pixel). All the first and the second order spatial derivatives are approximated by centered differences, but the temporal derivatives are approximated by forward differences, resulting in an explicit scheme to solve $O_m(k\Delta t)$ and $\rho_m(k\Delta t)$ successively, where Δt is the temporal stepsize. However, the explicit scheme is only conditionally stable: if Δt is too large, we cannot obtain solutions with controlled error. As we have not investigated this issue thoroughly, we simply borrow the stability condition for two-dimensional standard heat equations $f_t = \kappa(f_{xx} + f_{yy})$:

$$\Delta t \leq \frac{1}{4\kappa} \min((\Delta x)^2, (\Delta y)^2).$$

In our problem, Δx and Δy are naturally chosen as 1, and the coefficients of $O_{xx} + O_{yy}$ and $\rho_{xx} + \rho_{yy}$ are a_7 and b_7 , respectively. So the temporal stepsize should satisfy:

$$\Delta t \leq \frac{1}{4 \max(a_7, b_7)}. \quad (4)$$

If we find that the above condition is violated during optimization at a time step k , we will break Δt into smaller temporal stepsizes $\delta t = \Delta t/K$, such that it satisfies condition (4), and compute $O_m(k \cdot \Delta t + i \cdot \delta t)$, $i = 1, 2, \dots, K$, successively, using interpolated values of $a_j(t)$ and $b_j(t)$, until we obtain $O_m((k+1)\Delta t)$. We have found that this method works for all our experiments.

2.3.2 Initial functions of O and ρ

Good initialization can enhance the approximation accuracy of the learnt PDEs. In our current implementation, we simply set the initial functions of O and ρ as the input image:

$$O_m(\mathbf{x}, 0) = \rho_m(\mathbf{x}, 0) = I_m(\mathbf{x}), \quad m = 1, 2, \dots, M.$$

However, this is not the only choice. If the user has some prior knowledge about the input/output mapping, s/he can choose better initial functions. We *deliberately* do not do so in order to show the learning power of our framework.

2.3.3 Initialization of $\{a_i\}$ and $\{b_i\}$

We initialize $\{a_i(t)\}$ successively in time while fixing $b_i(t) \equiv 0, i = 0, 1, \dots, 16$. At the first time step, without any prior information, $O_m(\Delta t)$ is expected to be $\Delta t \tilde{O}_m + (1 - \Delta t)O_m(0)$. So $\partial O_m / \partial t|_{t=\Delta t}$ is expected to be $\tilde{O}_m - O_0$ and we may solve $\{a_i(0)\}$ such that the difference

$$s(\{a_i(0)\}) \equiv \frac{1}{2} \sum_{m=1}^M \int_{\Omega} \left[\sum_{j=0}^{16} a_j(0) \text{inv}_j(\rho_m(0), O_m(0)) - (\tilde{O}_m - O_0) \right]^2 d\Omega$$

between the left and the right hand sides of (1) is minimized, where the integration here should be understood as summation. After solving $\{a_i(0)\}$, we can have $O_m(\Delta t)$ by solving (1) at $t = \Delta t$. Suppose at the $(k + 1)$ -th step, we have solved $O_m(k\Delta t)$, then we may expect that $O_m((k + 1)\Delta t) = \frac{\Delta t}{1 - k\Delta t} \tilde{O}_m + \frac{(1 - k\Delta t) - \Delta t}{1 - k\Delta t} O_m(k\Delta t)$ so that $O_m((k + 1)\Delta t)$ could move directly towards \tilde{O}_m . So $\partial O_m / \partial t|_{t=(k+1)\Delta t}$ is expected to be $\frac{1}{1 - k\Delta t} [\tilde{O}_m - O_m(k\Delta t)]$ and $\{a_i(k\Delta t)\}$ can be solved in the same manner as $\{a_i(0)\}$.

2.3.4 Choice of Other Parameters

As it does not seem to have a systematic way to estimate the optimal values of other parameters, we simply fix their values as: $M = 60, \Delta t = 0.05$ and $\lambda_i = \mu_i = 10^{-7}, i = 0, \dots, 16$.

3 Experimental Results

In this section, we apply our framework to design PDEs for five basic image processing problems: blur, edge detection, denoising, deblurring and segmentation. For each problem, we prepare sixty 150×150 images and their ground truth outputs as training image pairs. After the PDE system is learnt, we apply it to testing images. The goal of these experiments is to show the beauty of our framework: the same approach for different problems and the performance of learnt PDEs is comparable to those problem-specific methods. We believe that if problem-specific techniques are also used, e.g. using better initial functions according to some prior knowledge of the problem, the performance of learnt PDEs can be even better.



Figure 1: Partial results of image blurring. The top row are the input images. The middle row are the outputs of our learnt PDEs. The bottom row are the ground truth images obtained by blurring the input image with a Gaussian kernel. One can see that the output of our PDEs is visually indistinguishable from the ground truth.

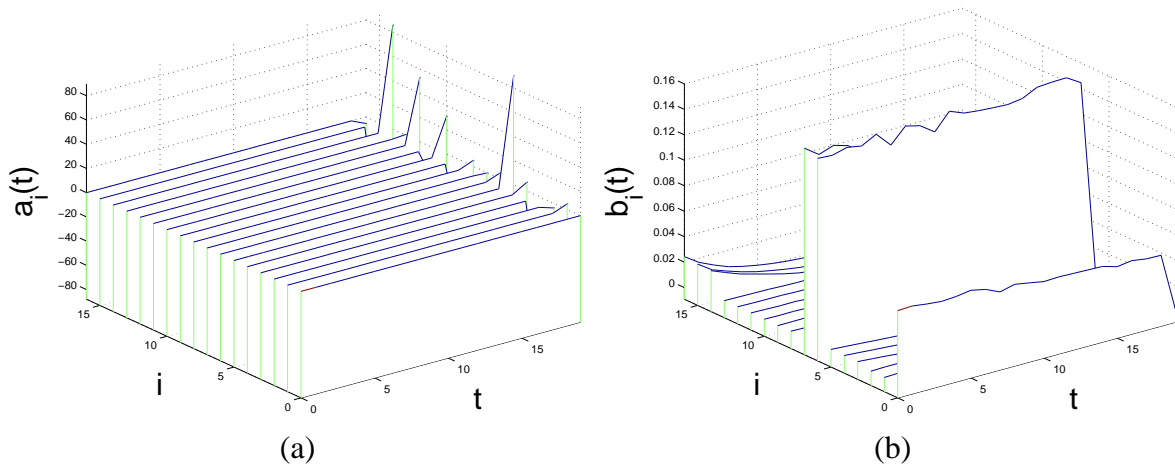


Figure 2: The learnt coefficients (a) a_i and (b) b_i , $i = 0, \dots, 16$, for image blurring.

Blurring. For image blurring (Figure 1), the output image is expected to be the convolution of the input image with a 5×5 Gaussian kernel with $\sigma = 1$. It is well known [11] that this corresponds to evolving with the standard heat equation $O_t = \eta(O_{xx} + O_{yy})$. One can see from Figure 1 that our learnt PDEs produce outputs that are visually indistinguishable from the ground truth. Actually, the average per pixel error, measured in root mean squared error, on the testing images is only 0.46 graylevels. To see whether the learnt PDEs are close to the ground truth PDEs, we plot the curves of the learnt coefficients in Figure 2. One can see that the learnt PDEs are *not* close to the standard heat equation, which corresponds to $a_7 = \text{const} > 0$, $a_i \equiv 0$, $i \neq 7$, and $b_j \equiv 0$, $j = 0, \dots, 16$. This trivial experiment shows that:

1. the learnt PDEs may be different from the true PDEs that an information process obeys;
2. the learnt PDEs could be effective to produce the expected results at time $t = 1$;
3. therefore it is unnecessary to retrieve the true PDEs that an information process really obeys (which may be extremely hard) in order to obtain the desired outputs².

These observations justify the usefulness of learning-based PDEs.

Edge Detection. For image edge detection (Figure 3), the well-known Canny detector [6] usually outputs an edge map that includes all edge pixels that are relatively strong within a certain neighborhood. Hence the resulting edge maps often contain minute edges that may be visually insignificant. We hope that we can obtain PDEs that output visually salient edges only. To this end, we select 2 or 3 images from each category of Corel photo library CDs, where 1 or 2 of them are collected for training and the remaining 1 or 2 images are for testing. The preparation of the visually salient edge maps is assisted by the edge detector in [3]³. We first use it to generate relatively rich edge maps and then manually delete visually

²In theory, for different mappings there must be inputs such that the outputs of the mappings are different. However, as natural images are highly structured, they only account for a tiny portion of the Euclidean space whose dimension is the number of pixels in the images. So it is possible that different mappings are more or less identical on such a tiny subset of the whole space.

³Code available at <http://www4.comp.polyu.edu.hk/~cslzhang/>

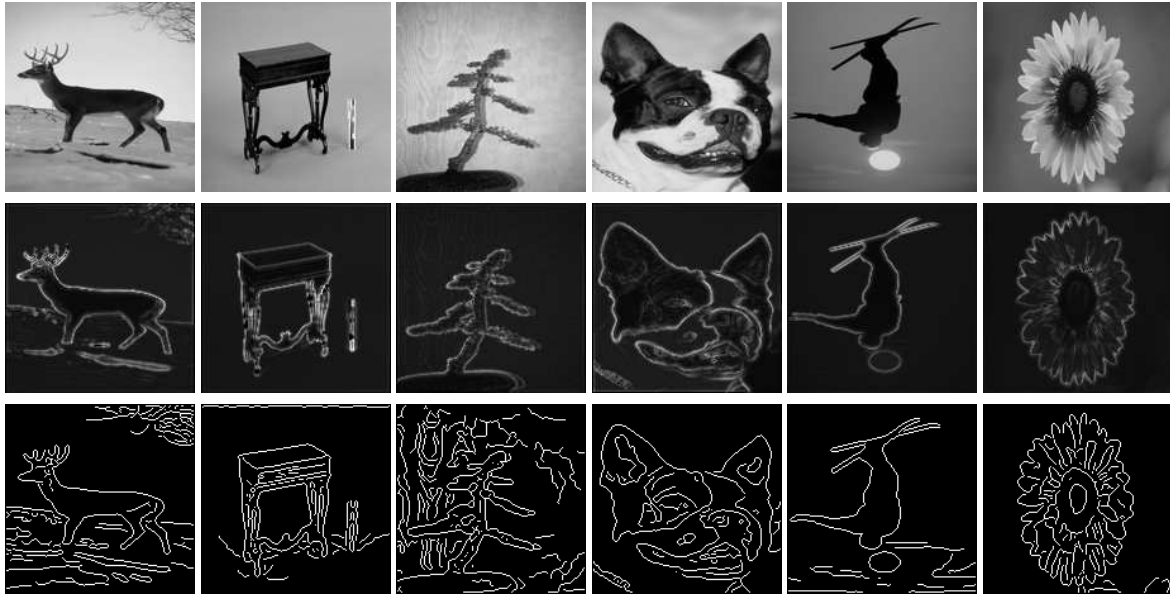


Figure 3: Partial results of edge detection. The top row are the input images. The middle row are the outputs of our learnt PDEs. The bottom row are the edge maps by the Canny detector [6].

insignificant edges. In this way, we obtain 60 training image pairs. Figure 3 shows part of the results on the collected 60 testing images. One can see that our PDEs respond selectively to edges and basically produce visually significant edges, while the edge maps of the Canny detector (using the function in Matlab) are more chaotic. Note that the solution of our PDEs is supposed to be more or less smooth functions. So one cannot expect that our PDEs produce exactly binary edge maps. Rather, an approximation of binary edge maps is produced.

The curves of the learnt coefficients are shown in Figure 4. Currently we are unable to analyze the obtained PDEs in detail as this work seems to be non-trivial. So we leave the analysis to future work.

Denoising. For image denoising (Figure 5), we generate input images by adding zero-mean Gaussian white noise, with $\sigma = 25$, to the original noiseless images and use the original images as the ground truth. One can see from Figure 5 that our PDEs suppress most of the noise while preserving the edges well. On the testing images, the PSNRs of our PDEs are in

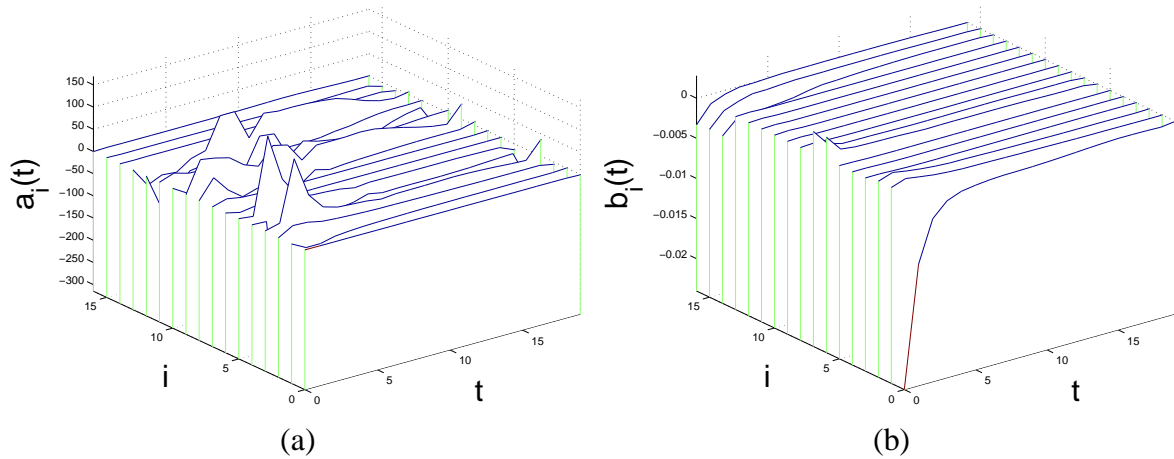


Figure 4: The learnt coefficients (a) a_i and (b) b_i , $i = 0, \dots, 16$, for edge detection.



Figure 5: Partial results of image denoising. The top row are the input noisy images. The second row are the outputs of our learnt PDEs. The third row are the outputs by the ramp preserving complex diffusion method [10] (using the original code by the authors, with parameters being tuned such that the average PSNR is the highest). The fourth row are the outputs of the BLS-GSM method [24] (using the original code by the authors).

the range of $27.87 \pm 2.07\text{dB}$, while those of a state-of-the-art PDE based method [10]⁴ are in $26.91 \pm 2.68\text{dB}$. So we easily obtain PDEs that produce results comparable with those by [10], which was designed with a lot of intelligence. We also compare our results with those of the BLS-GSM method [24]⁵, which is considered one of the best denoising algorithms. Being fed with the known noise level, their PSNRs are in $28.87 \pm 2.54\text{dB}$. Although their PSNRs are higher than ours, their denoised images exhibit ringing artifacts along strong edges. So our PDEs are indeed quite satisfactory.

The curves of the learnt coefficients are shown in Figure 6.

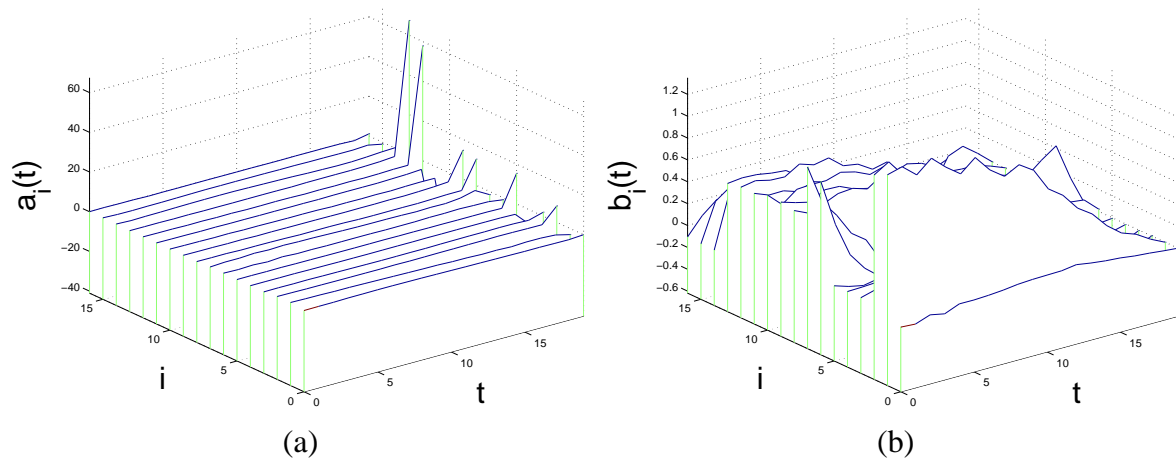


Figure 6: The learnt coefficients (a) a_i and (b) b_i , $i = 0, \dots, 16$, for denoising.

Deblurring. For image deblurring (Figure 7), we generate input images by blurring high resolution images using a Gaussian kernel with $\sigma = 1$. The original images are used as the ground truth. One can see from Figure 7 that our PDEs produce sharp images out of the blurred images. We also present the results of [28] and [27]⁶, which are state-of-the-art PDE and non-PDE-based image deblurring algorithms, respectively. The PSNR of our PDEs is in the range of $34.68 \pm 3.46\text{dB}$, while those of [28] and [27] are $29.46 \pm 4.07\text{dB}$ and $27.82 \pm 3.92\text{dB}$, respectively. So the performance of our learnt PDEs is comparable to theirs.

⁴Code available at <http://www.math.ucla.edu/~gilboa/PDE-filt/diffusions.html>

⁵Code available at <http://decsai.ugr.es/~javier/denoise/>

⁶Code available at <http://www.soe.ucsc.edu/~htakeda/AKTV>

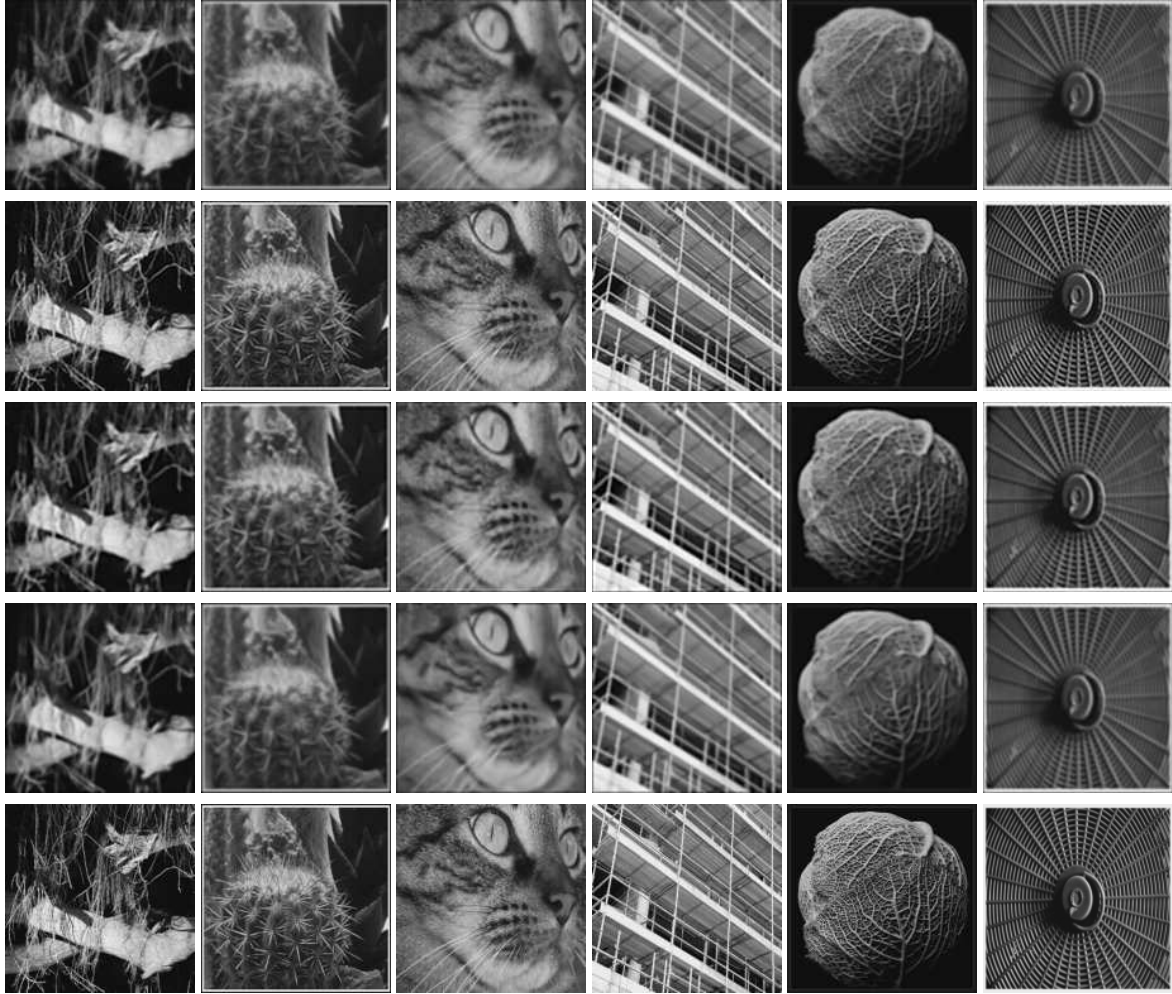


Figure 7: Partial results of image deblurring. The top row are the input images. The second row are the output of our learnt PDEs. The third row are the results of a PDE-based deblurring algorithm [28]. The fourth row are the results of a recent non-PDE-based algorithm [27] (using the original code by the authors). The last row are the ground truth images.

The curves of the learnt coefficients are shown in Figure 8.

Segmentation. For image segmentation, it is a highly ill posed problem and there are many criteria to define the goal of segmentation, e.g., breaking images into regions with similar intensity, color, texture, or expected shape. As none of current image segmentation algorithms can perform object level segmentation out of complex background well, we choose to require our PDEs to achieve a reasonable goal, namely segmenting relatively darker objects against relatively simple backgrounds, where both the foreground and the background can be tex-

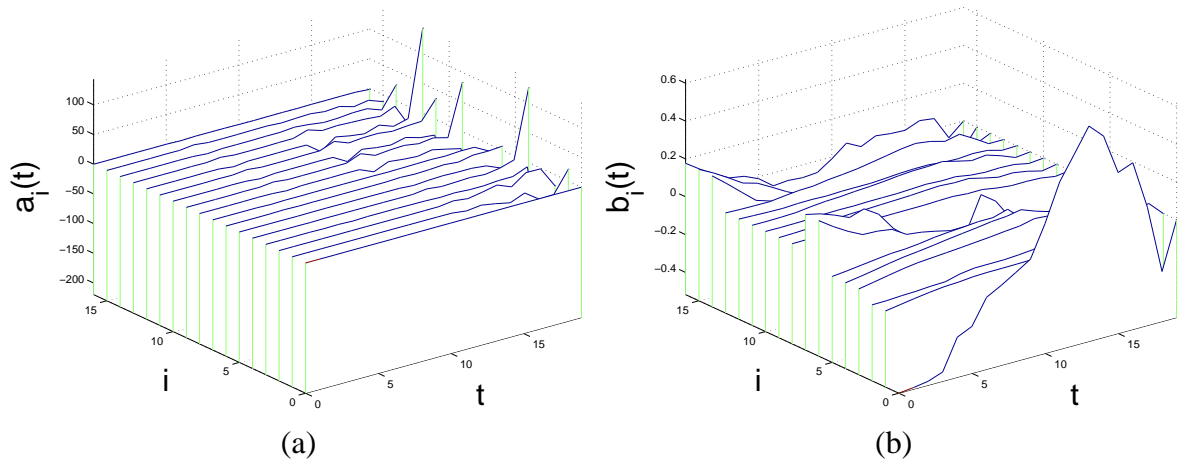


Figure 8: The learnt coefficients (a) a_i and (b) b_i , $i = 0, \dots, 16$, for deblurring.



Figure 9: Examples of the training images for image segmentation. In each group of images, on the left is the input image and on the right is the ground truth output mask.



Figure 10: Partial results of image segmentation. The top row are the input images. The second row are the masks obtained by thresholding the mask maps output by our learnt PDEs with a *constant* threshold 0.5. The third row are the segmentation results of active contour [15]. The bottom row are the results of normalized cut [26]. The results in the last two rows are produced using the original code by the authors.

tured and simple thresholding *cannot* separate them. So we select 60 images with relatively darker foregrounds and relatively simple backgrounds, but the foreground is *not* of uniformly lower graylevels than the background, and prepare the manually segmented binary masks as the outputs of the training images (Figure 9). Part of the segmentation results are shown in Figure 10, where we have thresholded the output mask maps of our learnt PDEs with a *constant* threshold 0.5. We see that our learnt PDEs produce fairly good object masks. We also test the active contour method by Li et al. [15]⁷ and the celebrated normalized cut method [26]⁸. One can see from Figure 10 that the active contour method [15] cannot segment object details due to its smoothness constraint on the object shape and the normalized cut method [26] cannot produce a closed foreground region. To provide quantitative evaluation, we use the F -measure that merges the precision and recall of segmentation:

$$F_\alpha = \frac{(1 + \alpha) \cdot recall \cdot precision}{\alpha \cdot precision + recall}, \text{ where}$$

$$recall = \frac{|A \cap B|}{|A|}, \quad precision = \frac{|A \cap B|}{|B|},$$

in which A is the ground truth mask, B is the computed mask and $|\cdot|$ denotes the area of a region. The most common choice of α is 2. On our testing images, the F_2 measures of our PDEs, [15] and [26] are 0.90 ± 0.05 , 0.83 ± 0.16 and 0.61 ± 0.20 , respectively. One can see that the performance of our PDEs is better than theirs, in both visual quality and quantitative measure.

The curves of the learnt coefficients are shown in Figure 11.

We also present the evolution process of the mask maps across time (Figure 12). One can see that although the foreground is relatively *darker* than the background, the PDEs correctly detect the most salient points/edges and then propagate the information across the foreground region, resulting in a *brighter* output region for the foreground.

⁷Code available at http://www.engr.uconn.edu/~cmli/code/LevelSet_ChunmingLi_v1.rar

⁸Code available at <http://www.cis.upenn.edu/~jshi/software/>

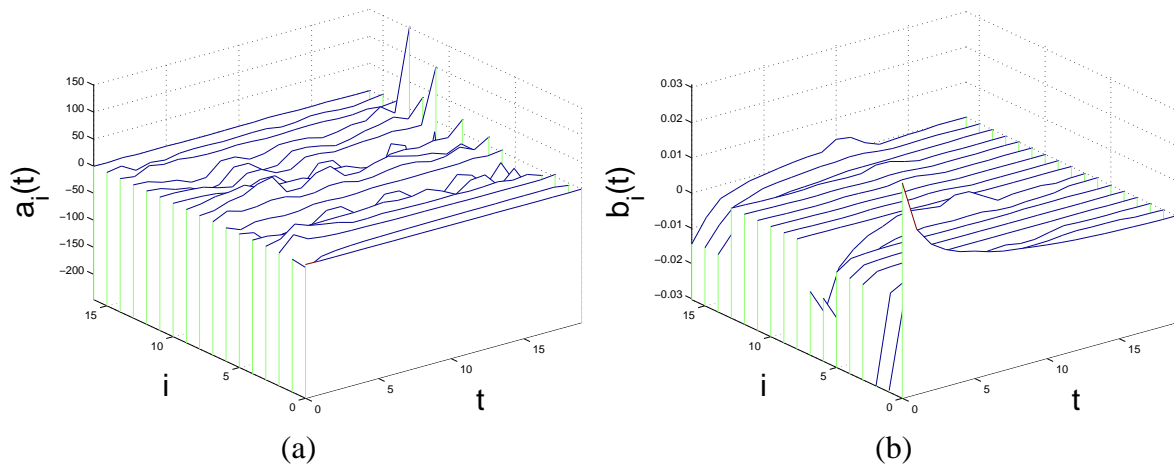


Figure 11: The learnt coefficients (a) a_i and (b) b_i , $i = 0, \dots, 16$, for image segmentation.

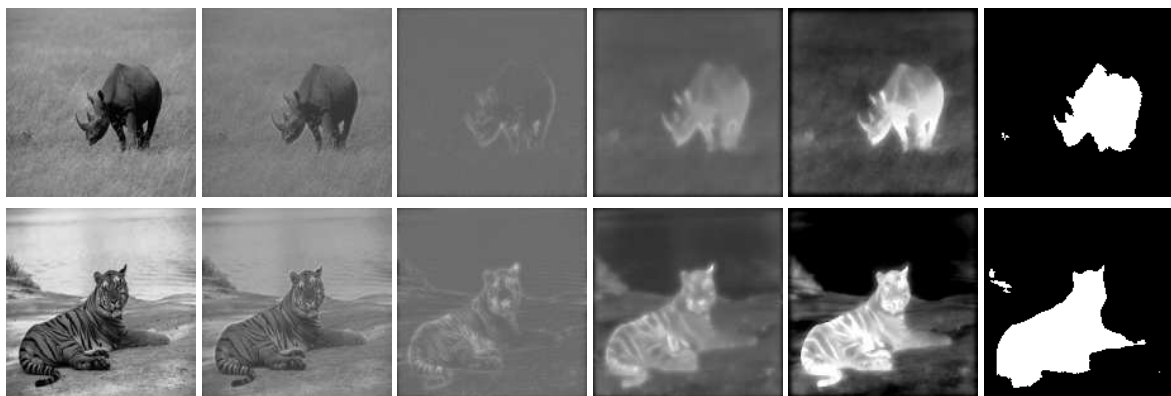


Figure 12: The evolution of the mask maps. For each row, the first image is the input image, the second to the fifth are the mask maps at time $t = 0.25, 0.50, 0.75, 1.0$, respectively, and the last image is the final mask map thresholded at 0.5.

4 Conclusions and Future Work

In this paper, we have presented a framework of learning PDEs from examples for image processing problems. The experimental results on some image processing problems show that our theory is promising. However, the current work is still preliminary; we plan to push our research in the following directions.

First, we will apply our framework to more image processing/computer vision problems to find out to what extent it works. We do find some problems, e.g., image inpainting, that our current framework does not work well. But we also want to find more problems that our current framework works well. This would help us understand the level of complexity of image processing/computer vision problems. Second, the time required to learn the PDEs is a bit long. On our 2.4GHz Duro Core Pentium PC, the time is about 3 days, using our unoptimized codes. Note that such a time frame may still be shorter than that for a human to design PDEs as effective for the same problem. Although the computation can easily be parallelized, better mechanisms, e.g., better initialization of the combination coefficients and real-time optimal control techniques [5], should be explored. Third, although hard, the theoretical analysis on our approach, e.g., the well-posedness of the learnt PDEs and the numerical stability condition for the temporal stepsize Δt , should be tried. Fourth, it is attractive to analyze the learnt PDEs and find out their connections with the biological vision; we hope to borrow some ideas from the biological vision. We expect that someday learning-based PDEs, in their improved formulations, could be a general framework for designing PDEs for most image processing problems.

References

- [1] Alvarez, L., Guichard, F., Lions, P.L., Morel, J.M.: Axioms and fundamental equations of image processing. *Arch. for Rational Mechanics* **123**(3), 199–257 (1993)
- [2] Aubert, G., Kornprobst, P.: *Mathematical Problems in Image Processing*. Springer-Verlag (2002)

- [3] Bao, P., Zhang, L., Wu, X.: Canny edge detection enhancement by scale multiplication. *IEEE Trans. Pattern Analysis and Machine Intelligence* **27**(9), 1485–1490 (2005)
- [4] Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: *SIGGRAPH*, pp. 417–424 (2000)
- [5] Biegler, L., et al.: *Real-Time PDE-Constrained Optimization*. SIAM (2007)
- [6] Canny, J.: A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**, 679–714 (1986)
- [7] Cao, F.: *Geometric Curve Evolution and Image Processing*. Lecture Notes in Mathematics, No. 1805. Springer-Verlag (2003)
- [8] Caselles, V., Morel, J.M., Sapiro, G., A. Tannenbaum (*eds.*): Special issue on partial differential equations and geometry-driven diffusion in image processing and analysis. *IEEE Trans. Image Processing* **7**(3) (1998)
- [9] Gabor, D.: Information theory in electron microscopy. *Laboratory Investigation* **14**, 801–807 (1965)
- [10] Gilboa, G., Sochen, N., Zeevi, Y.: Image enhancement and denoising by complex diffusion processes. *IEEE Trans. Pattern Analysis and Machine Intelligence* **26**(8), 1020–1036 (2004)
- [11] ter Haar Romeny, B.M.: *Geometry-Driven Diffusion in Computer Vision*. Kluwer Academic Publishers (1994)
- [12] Jain, A.: Partial differential equations and finite-difference methods in image processing, part 1. *J. Optimization Theory and Applications* **23**, 65–91 (1977)
- [13] Kimia, B., Tannenbaum, A., Zucker, S.: On optimal control methods in computer vision and image processing. In B. ter Haar Romeny (ed.) *Geometry-Driven Diffusion in Computer Vision*, Kluwer Academic Publishers, 1994

- [14] Koenderink, J.: The structure of images. *Biological Cybernetics* **50**, 363–370 (1984)
- [15] Li, C., Xu, C., Gui, C., Fox, M.: Level set evolution without re-initialization: A new variational formulation. In: *Proc. Computer Vision and Pattern Recognition* (2005)
- [16] Lin, Z., Zhang, W., Tang, X.: Learning partial differential equations for computer vision (2008). Microsoft Technical Report #MSR-TR-2008-189
- [17] Lions, J.L.: *Optimal Control Systems Governed by Partial Differential Equations*. Springer-Verlag (1971)
- [18] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. *IEEE Trans. Pattern Analysis and Machine Intelligence* **27**(10), 1615–1630 (2005)
- [19] Olver, P.: *Applications of Lie Groups to Differential Equations*. Springer-Verlag, New York (1993)
- [20] Osher, S., Rudin, L.I.: Feature-oriented image enhancement using shock filters. *SIAM J. Numerical Analysis* **27**(4), 919–940 (1990)
- [21] Papadakis, N., Corpetti, T., Mémin, E.: Dynamically consistent optical flow estimation. In: *Proc. Intn'l Conf. Computer Vision* (2007)
- [22] Papadakis, N., Mémin, E.: Variational optimal control technique for the tracking of deformable objects. In: *Proc. Int. Conf. Computer Vision* (2007)
- [23] Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis and Machine Intelligence* **12**(7), 629–639 (1990)
- [24] Portilla, J., Strela, V., Wainwright, M., Simoncelli, E.: Image denoising using scale mixtures of Gaussians in the wavelet domain. *IEEE Trans. Image Processing* **12**(11), 1338–1351 (2003)
- [25] Sapiro, G.: *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press (2001)

- [26] Shi, J., Malik, J.: Normalized cuts and image segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(8), 888–905 (2000)
- [27] Takeda, H., Farsiu, S., Milanfar, P.: Deblurring using regularized locally adaptive kernel regression. *IEEE Trans. Image Processing* **17**(4), 550–563 (2008)
- [28] Welk, M., Theis, D., Brox, T., Weickert, J.: PDE-based deconvolution with forward-backward diffusivities and diffusion tensors. In: *Proc. Scale Space and PDE Methods in Computer Vision*, pp. 585–597 (2005)
- [29] Witkin, A.: Scale-space filtering. In: *Proc. Int. Joint Conf. Artificial Intelligence* (1983)

Appendix 1: Shift Invariance of PDEs

We prove that the coefficients a_j and b_j must be independent of \mathbf{x} .

Proof: We prove for L_O only. We may rewrite

$$L_O(\mathbf{a}(\mathbf{x}, t), \langle O \rangle, \langle \rho \rangle) = \tilde{L}_O(\langle O \rangle, \langle \rho \rangle, \mathbf{x}, t),$$

and it suffices to prove that \tilde{L}_O is independent of \mathbf{x} .

By the definition of shift invariance, when $I(\mathbf{x})$ changes to $I(\mathbf{x} - \mathbf{x}_0)$ by shifting with a displacement \mathbf{x}_0 , $O(\mathbf{x})$ and $\rho(\mathbf{x})$ will change to $O(\mathbf{x} - \mathbf{x}_0)$ and $\rho(\mathbf{x} - \mathbf{x}_0)$, respectively. So the pair $(\rho(\mathbf{x} - \mathbf{x}_0), O(\mathbf{x} - \mathbf{x}_0))$ fulfils (1), i.e.,

$$\begin{aligned} \frac{\partial O(\mathbf{x} - \mathbf{x}_0)}{\partial t} &= \tilde{L}_O(\langle O(\mathbf{x} - \mathbf{x}_0) \rangle, \langle \rho(\mathbf{x} - \mathbf{x}_0) \rangle, \mathbf{x}, t) \\ &= \tilde{L}_O(\langle O \rangle(\mathbf{x} - \mathbf{x}_0), \langle \rho \rangle(\mathbf{x} - \mathbf{x}_0), \mathbf{x}, t). \end{aligned} \quad (5)$$

Next, we replace $\mathbf{x} - \mathbf{x}_0$ in the above equation with \mathbf{x} and have:

$$\frac{\partial O(\mathbf{x})}{\partial t} = \tilde{L}_O(\langle O \rangle(\mathbf{x}), \langle \rho \rangle(\mathbf{x}), \mathbf{x} + \mathbf{x}_0, t). \quad (6)$$

On the other hand, the pair $(\rho(\mathbf{x}), O(\mathbf{x}))$ also fulfils (1), i.e.,

$$\frac{\partial O(\mathbf{x})}{\partial t} = \tilde{L}_O(\langle O \rangle(\mathbf{x}), \langle \rho \rangle(\mathbf{x}), \mathbf{x}, t). \quad (7)$$

Therefore,

$$\tilde{L}_O(\langle O \rangle(\mathbf{x}), \langle \rho \rangle(\mathbf{x}), \mathbf{x} + \mathbf{x}_0, t) = \tilde{L}_O(\langle O \rangle(\mathbf{x}), \langle \rho \rangle(\mathbf{x}), \mathbf{x}, t),$$

$\forall \mathbf{x}_0$ that confines the input image inside Ω .

So \tilde{L}_O is independent of \mathbf{x} .

□

Appendix 2: The Gâteaux Derivatives

We compute the Gâteaux derivatives by perturbation. The underlying theory can be found in [16, 21, 22]. The Lagrangian function is:

$$\begin{aligned} & \tilde{J}(\{O_m\}_{m=1}^M, \{a_j\}_{j=0}^{16}, \{b_j\}_{j=0}^{16}; \{\varphi_m\}_{m=1}^M, \{\phi_m\}_{m=1}^M) \\ = & J(\{O_m\}_{m=1}^M, \{a_j\}_{j=0}^{16}, \{b_j\}_{j=0}^{16}) \\ & + \sum_{m=1}^M \int_Q \varphi_m [(O_m)_t - L_O(\mathbf{a}, \langle O_m \rangle, \langle \rho_m \rangle)] \mathbf{d}Q \\ & + \sum_{m=1}^M \int_Q \phi_m [(\rho_m)_t - L_\rho(\mathbf{b}, \langle \rho_m \rangle, \langle O_m \rangle)] \mathbf{d}Q, \end{aligned} \quad (8)$$

where φ_m and ϕ_m are the adjoint functions.

To find the adjoint equations for φ_m , we perturb L_O and L_ρ with respect to O . The perturbations can be written as follows:

$$\begin{aligned} & L_O(\mathbf{a}, \langle O + \varepsilon \cdot \delta O \rangle, \langle \rho \rangle) - L_O(\mathbf{a}, \langle O \rangle, \langle \rho \rangle) \\ = & \varepsilon \cdot \left(\frac{\partial L_O}{\partial O}(\delta O) + \frac{\partial L_O}{\partial O_x} \frac{\partial(\delta O)}{\partial x} + \dots + \frac{\partial L_O}{\partial O_{yy}} \frac{\partial^2(\delta O)}{\partial y^2} \right) + o(\varepsilon) \\ = & \varepsilon \sum_{p \in \wp} \frac{\partial L_O}{\partial O_p} \frac{\partial^{|p|}(\delta O)}{\partial p} + o(\varepsilon) \\ = & \varepsilon \sum_{p \in \wp} \sigma_{O;p} \frac{\partial^{|p|}(\delta O)}{\partial p} + o(\varepsilon), \end{aligned} \quad (9)$$

$$\begin{aligned} & L_\rho(\mathbf{b}, \langle \rho \rangle, \langle O + \varepsilon \cdot \delta O \rangle) - L_\rho(\mathbf{b}, \langle \rho \rangle, \langle O \rangle) \\ = & \varepsilon \sum_{p \in \wp} \sigma_{\rho;p} \frac{\partial^{|p|}(\delta O)}{\partial p} + o(\varepsilon), \end{aligned}$$

where

$$\begin{aligned}
\wp &= \{\emptyset, x, y, xx, xy, yy\}, \\
|p| &= \text{the length of string } p, \\
\sigma_{O;p} &= \frac{\partial L_O}{\partial O_p} = \sum_{i=0}^{16} a_i \frac{\partial \text{inv}_i(\rho, O)}{\partial O_p}, \quad \text{and} \\
\sigma_{\rho;p} &= \frac{\partial L_\rho}{\partial O_p} = \sum_{i=0}^{16} b_i \frac{\partial \text{inv}_i(O, \rho)}{\partial O_p}.
\end{aligned}$$

Then the difference in \tilde{J} caused by perturbing O_k only is

$$\begin{aligned}
&\delta \tilde{J}_k \\
&= \tilde{J}(\dots, O_k + \varepsilon \cdot \delta O_k, \dots) - \tilde{J}(\dots, O_k, \dots) \\
&= \frac{1}{2} \int_{\Omega} \left[\left((O_k + \varepsilon \cdot \delta O_k)(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \right)^2 - \left(O_k(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \right)^2 \right] d\Omega \\
&\quad + \int_Q \varphi_k \{ [(O_k + \varepsilon \cdot \delta O_k)_t - L_O(\mathbf{a}, \langle O_k + \varepsilon \cdot \delta O_k \rangle, \langle \rho_k \rangle)] - [(O_k)_t - L_O(\mathbf{a}, \langle O_k \rangle, \langle \rho_k \rangle)] \} dQ \\
&\quad + \int_Q \phi_k \{ [(\rho_k)_t - L_\rho(\mathbf{b}, \langle \rho_k \rangle, \langle O_k + \varepsilon \cdot \delta O_k \rangle)] - [(\rho_k)_t - L_\rho(\mathbf{b}, \langle \rho_k \rangle, \langle O_k \rangle)] \} dQ \\
&= \varepsilon \int_{\Omega} \left(O_k(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \right) \delta O_k(\mathbf{x}, 1) d\Omega + \varepsilon \int_Q \varphi_k (\delta O_k)_t dQ \\
&\quad - \varepsilon \int_Q \varphi_k \sum_{p \in \wp} \sigma_{O;p} \frac{\partial^{|p|}(\delta O_k)}{\partial p} dQ - \varepsilon \int_Q \phi_k \sum_{p \in \wp} \sigma_{\rho;p} \frac{\partial^{|p|}(\delta O_k)}{\partial p} dQ + o(\varepsilon).
\end{aligned} \tag{10}$$

As the perturbation δO_k should satisfy that

$$\delta O_k|_{\Gamma} = 0 \quad \text{and} \quad \delta O_k|_{t=0} = 0,$$

due to the boundary and initial conditions of O_k , if we assume that

$$\varphi_k|_{\Gamma} = 0,$$

then integrating by parts, the integration on the boundary Γ will vanish. So we have

$$\begin{aligned}
& \delta \tilde{J}_k \\
= & \varepsilon \int_{\Omega} \left(O_k(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \right) \delta O_k(\mathbf{x}, 1) d\Omega \\
& + \varepsilon \int_{\Omega} (\varphi_k \cdot \delta O_k)(\mathbf{x}, 1) d\Omega - \varepsilon \int_Q (\varphi_k)_t \delta O_k dQ \\
& - \varepsilon \int_Q \sum_{p \in \wp} (-1)^{|p|} \frac{\partial^{|p|} (\sigma_{O;p} \varphi_k)}{\partial p} \delta O_k dQ \\
& - \varepsilon \int_Q \sum_{p \in \wp} (-1)^{|p|} \frac{\partial^{|p|} (\sigma_{\rho;p} \phi_k)}{\partial p} \delta O_k dQ + o(\varepsilon) \\
= & \varepsilon \int_Q \left[\left(\varphi_k + O_k(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \right) \delta(t-1) \right. \\
& \left. - (\varphi_k)_t - (-1)^{|p|} \frac{\partial^{|p|} (\sigma_{O;p} \varphi_k + \sigma_{\rho;p} \phi_k)}{\partial p} \right] \delta O_k dQ + o(\varepsilon).
\end{aligned} \tag{11}$$

By letting $\varepsilon \rightarrow 0$, we have that the adjoint equation for φ_k is

$$\left\{ \begin{array}{l} \frac{\partial \varphi_m}{\partial t} + \sum_{p \in \wp} (-1)^{|p|} (\sigma_{O;p} \varphi_m + \sigma_{\rho;p} \phi_m)_p = 0, \quad (\mathbf{x}, t) \in Q, \\ \varphi_m = 0, \quad (\mathbf{x}, t) \in \Gamma, \\ \varphi_m|_{t=1} = \tilde{O}_m - O_m(1), \quad \mathbf{x} \in \Omega, \end{array} \right. \tag{12}$$

in order that $\frac{\partial \tilde{J}}{\partial O_k} = 0$. Similarly, the adjoint equation for ϕ_k is

$$\left\{ \begin{array}{l} \frac{\partial \phi_m}{\partial t} + \sum_{p \in \wp} (-1)^{|p|} (\tilde{\sigma}_{O;p} \varphi_m + \tilde{\sigma}_{\rho;p} \phi_m)_p = 0, \quad (\mathbf{x}, t) \in Q, \\ \phi_m = 0, \quad (\mathbf{x}, t) \in \Gamma, \\ \phi_m|_{t=1} = 0, \quad \mathbf{x} \in \Omega, \end{array} \right. \tag{13}$$

where

$$\begin{aligned}
\tilde{\sigma}_{O;p} &= \frac{\partial L_O}{\partial \rho_p} = \sum_{i=0}^{16} a_i \frac{\partial \text{inv}_i(\rho, O)}{\partial \rho_p}, \quad \text{and} \\
\tilde{\sigma}_{\rho;p} &= \frac{\partial L_\rho}{\partial \rho_p} = \sum_{i=0}^{16} b_i \frac{\partial \text{inv}_i(O, \rho)}{\partial \rho_p}.
\end{aligned} \tag{14}$$

Also by perturbation, it is easy to check that:

$$\begin{aligned}
\frac{\partial \tilde{J}}{\partial a_i} &= \lambda_i a_i - \int_{\Omega} \sum_{m=1}^M \varphi_m \text{inv}_i(\rho_m, O_m) d\Omega, \\
\frac{\partial \tilde{J}}{\partial b_i} &= \mu_i b_i - \int_{\Omega} \sum_{m=1}^M \phi_m \text{inv}_i(O_m, \rho_m) d\Omega.
\end{aligned}$$

The above are also the Gâteaux derivatives of J with respect to a_i and b_i , respectively.