Image and Vision Computing xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Toward designing intelligent PDEs for computer vision: An optimal control approach ☆

Risheng Liu [a,d], Zhouchen Lin [b,*], Wei Zhang [c], Kewei Tang [d], Zhixun Su [d]

[a] Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, China
[b] Key Lab. of Machine Perception (MOE), School of EECS, Peking University, Beijing, China
[c] Department of Information Engineering, The Chinese University of Hong Kong, China
[d] School of Mathematical Sciences, Dalian University of Technology, Dalian, China

ARTICLE INFO

Article history:
Received 6 September 2011
Received in revised form 13 August 2012
Accepted 21 September 2012
Available online xxxx

Keywords:
Optimal control
PDEs
Computer vision
Image processing

ABSTRACT

Many computer vision and image processing problems can be posed as solving partial differential equations (PDEs). However, designing a PDE system usually requires high mathematical skills and good insight into the problems. In this paper, we consider designing PDEs for various problems arising in computer vision and image processing in a lazy manner: *learning PDEs from training data via an optimal control approach*. We first propose a general intelligent PDE system which holds the basic translational and rotational invariance rule for most vision problems. By introducing a PDE-constrained optimal control framework, it is possible to use the training data resulting from multiple ways (ground truth, results from other methods, and manual results from humans) to learn PDEs for different computer vision tasks. The proposed optimal control based training framework aims at learning a PDE-based regressor to approximate the unknown (and usually nonlinear) mapping of different vision tasks. The experimental results show that the learnt PDEs can solve different vision problems reasonably well. In particular, we can obtain PDEs not only for problems that traditional PDEs work well but also for problems that PDE-based methods have never been tried before, due to the difficulty in describing those problems in a mathematical way.

1. Introduction

The wide applications of partial differential equations (PDEs) in computer vision and image processing can be attributed to two main factors [1]. First, PDEs in classical mathematical physics are powerful tools to describe, model, and simulate many dynamics such as heat flow, diffusion, and wave propagation. Second, many variational problems or their regularized counterparts can often be effectively solved from their Euler–Lagrange equations. Therefore, in general there are two types of methods for designing PDEs for vision tasks. For the first kind of methods, PDEs are written down directly based on some understandings on the properties of mathematical operators or the physical natures of the problems (e.g., anisotropic diffusion [2], shock filter [3] and curve-evolution-based equations [4]). The second kind of methods basically defines an energy functional and then derive the evolution equations by computing the Euler–Lagrange equation of the energy functional (e.g., total-variation-based variational methods [5–7]). In either way, people have to heavily rely on their intuition on the vision tasks. Therefore, traditional PDE-based methods require good mathematical skills when choosing appropriate PDE forms and predicting the final effect of composing related operators such that the obtained PDEs roughly meet the goals. If people do not have enough intuition on a vision task, they may have difficulty in acquiring effective PDEs. Therefore, current PDE design methods greatly limit the applications of PDEs to a wider and more complex scope. This motivates us to explore whether we can acquire PDEs that are less artificial yet more powerful. In this paper, we give an affirmative answer to this question. We demonstrate that learning particular coefficients of a general intelligent PDE system from a given training data set might be a possible way of designing PDEs for computer vision in a lazy manner. Furthermore, borrowing this learning strategy from machine learning can generalize PDE techniques for more complex vision problems.

The key idea of our general intelligent PDE system is to assume that the PDEs in sought could be written as combinations of "atoms" which satisfy the general properties of vision tasks. As a preliminary investigation, we utilize the translational and rotational invariants as such "atoms" and propose the general intelligent PDE system as a linear combination of all these invariants[1] [8]. Then the

☆ This paper has been recommended for acceptance by Thomas Brox.

* Corresponding author. Tel.: +86 10 62753313.
E-mail addresses: rsliu0705@gmail.com (R. Liu), zlin@pku.edu.cn (Z. Lin), wzhang009@gmail.com (W. Zhang), tkwliaoning@gmail.com (K. Tang), zxsu@dlut.edu.cn (Z. Su).

[1] Currently, we only consider the case that the PDEs are linear combinations of fundamental differential invariants up to second order.

0262-8856/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.imavis.2012.09.004

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 [9], 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. Moreover, we also assume that the visual processing has two coupled evolutions: one controls the evolution of the output, and the other is for an indicator function that helps collect the global information to guide the evolution. Therefore, our general intelligent PDE system consists of two evolutionary PDEs. Both PDEs are coupled equations between the image and indicator, up to their second-order partial derivatives.

The theory of optimal control [10] has been well developed for over 50 years. With the enormous advances in computing power, optimal control is now widely used in multi-disciplinary applications such as biological systems, communication networks and socio-economic systems etc [11]. Optimal design and parameter estimation of systems governed by PDEs give rise to a class of problems known as PDE-constrained optimal control [9]. In this paper, a PDE-constrained optimal control technique as the training tool is introduced for our PDE system. We further propose a general framework for learning PDEs to accomplish a specific vision task via PDE-constrained optimal control, where the objective functional is to minimize the difference between the expected outputs and the actual outputs of the PDEs, given the input images. Such input–output image pairs are provided in multiple ways (e.g., ground truth, results from other methods or manually generated results by humans) for different tasks. Therefore, we can train the general intelligent PDE system to solve various vision problems which traditional PDEs may find difficult or even impossible, due to their difficulty in mathematical description. In summary, our contributions are as follows:

1. Our intelligent PDE system provides a new way to design PDEs for computer vision. Based on this framework, we can design particular PDEs for different vision tasks using different sources of training images. This may be very difficult for traditional PDE design methods. However, we would like to remind the readers that we have no intention to beat all the existing approaches for each task, because these approaches have been carefully and specially tuned for the task.
2. We propose a general data-based optimal control framework for training the PDE system. Fed with pairs of input and output images, the proposed PDE-constrained optimal control training model can automatically learn the combination coefficients in the PDE system. Unlike previous design methods, our approach requires much less human wits and can solve more difficult problems in computer vision.
3. We extend our basic framework for learning a system of PDEs for vector-valued images. With this framework, the correlation among different channels of images can be automatically (but implicitly) modeled. To further improve the learning ability of our framework, we also propose using multi-layer PDE systems. We show that our framework, as a general tool, is powerful in handling complex vision tasks with vector-valued images.

The rest of the paper is structured as follows. We first introduce in Section 2 our intelligent PDE system. In Section 3 we utilize the PDE-constrained optimal control technique as the training framework for our intelligent PDE system. In Section 4, we show two possible ways of extending our basic framework to handle more complicated vision problems with vector-valued images. Section 5 discusses the relationships between our work and the previous research. Then in Section 6 we evaluate our intelligent PDE system with optimal control training framework by a series of computer vision and image processing problems. Finally, we give concluding remarks and a discussion on the future work in Section 7.

Table 1
Notations.

Ω	An open bounded region in $\mathbb{R}^2$	$\partial\Omega$	Boundary of Ω
(x,y)	$(x,y)\in\Omega$, spatial variable	t	$t\in(0,T)$, temporal variable
Q	$\Omega\times(0,T)$	Γ	$\partial\Omega\times(0,T)$
$\lvert\cdot\rvert$	The area of a region	$\mathbf{X}^T$	Transpose of matrix (or vector)
$\lVert\cdot\rVert$	L^2 norm	$\mathrm{tr}(\cdot)$	Trace of matrix
∇u	Gradient of u	$\mathbf{H}_u$	Hessian of u
$\mathcal{P}$	$\mathcal{P}=\{(0,0),(0,1),(1,0),(0,2),(1,1),(2,0)\}$, index set for partial differentiation		

2. Intelligent PDE system

In this section, we introduce our intelligent PDE system and analyze its invariant property for computer vision problems.

2.1. General intelligent PDE system

We first assume that our PDEs should be 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. Furthermore, we also assume that the PDE system consists of two coupled PDEs. One is for the evolution of the output image u, and the other is for the evolution of an indicator function v. The goal of introducing the indicator function is to collect large-scale information in the image so that the evolution of u can be correctly guided. This idea is inspired by the edge indicator in [12] (page 193). So our PDE system can be written as:

$$\begin{cases} \dfrac{\partial u}{\partial t}-F_u(u,v,\mathbf{a})=0, & (x,y,t)\in Q,\\ u(x,y,t)=0, & (x,y,t)\in\Gamma,\\ u|_{t=0}=f_u, & (x,y)\in\Omega,\\ \dfrac{\partial v}{\partial t}-F_v(v,u,\mathbf{b})=0, & (x,y,t)\in Q,\\ v(x,y,t)=0, & (x,y,t)\in\Gamma,\\ v|_{t=0}=f_v, & (x,y)\in\Omega, \end{cases} \tag{1}$$

where Ω is the rectangular region occupied by the input image I, T is the time that the PDE system finishes the visual information processing and outputs the results, and f_u and f_v are the initial functions of u and v, respectively. The meaning of other notations in Eq. (1) can be found in Table 1. For computational issues and the ease of mathematical deduction, I will be padded with zeros of several pixels width around it. As we can change the unit of time, it is harmless to fix $T=1$. $\mathbf{a}=\{a_j\}$ and $\mathbf{b}=\{b_j\}$ are sets of functions defined on Q that are used to control the evolution of u and v, respectively.

2.2. Translationally and rotationally invariant PDE formulation

Although the motivations of different PDE models in computer vision vary, their formulations are similar, i.e., the most existing evolutionary PDEs for an image u can be brought to as follows[2]:

$$\frac{\partial u}{\partial t}-F(u,\nabla u,\mathbf{H}_u)=0, \tag{2}$$

[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.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

where F is a function of u, ∇u and $\mathbf{H}_u$. In this view, the differences between various PDE models lie in choosing different functions F in Eq. (2) such that the obtained PDEs roughly meet the goals. As the space of all these PDEs is infinitely dimensional, people have to heavily rely on their intuition (e.g., smoothness of edge contour and surface shading) on the vision task for designing the specific PDE formulation. Such intuition should be quantified and be described using the operators (e.g., gradient and Laplacian), functions (e.g., quadratic and square root functions) and numbers (e.g., 0.5 and 1) that people are familiar with. Therefore, the designed PDEs can only reflect very limited aspects of a vision task (hence are not robust in handling complex situations in real applications) and also appear rather artificial.

In our framework, we provide a new insight to design the PDE formulation in Eq. (1). Specifically, to find the right form for the PDE system, we start with the properties that our PDE system should have, in order to narrow down the search space. We notice that translationally and rotationally invariant properties are very important for computer vision, i.e., in most vision tasks, when the input image is translated or rotated, the output image is also translated or rotated by the same amount. So we require that our PDE system is translationally and rotationally invariant. According to the differential invariant theory [8], the form of our PDEs (i.e., F_u and F_v) must be functions of the fundamental differential invariants under the group of translation and rotation. The fundamental differential invariants are invariant under translation and rotation and other invariants can be written as their functions. We list those up to second order in Table 2, where some notations can be found in Table 1. In the sequel, we shall use $\{\mathrm{inv}_j(u,v)\}_{j=0}^{16}$ to refer to them in order. Note that those invariants are ordered with v going before u. We may reorder them with u going before v. In this case, the j-th invariant will be referred to as $\mathrm{inv}_j(v,u)$.

As ∇u and $\mathbf{H}_u$ change to $\mathbf{R}\nabla u$ and $\mathbf{R}\mathbf{H}_u\mathbf{R}^T$, respectively, when the image is rotated by a matrix $\mathbf{R}$, it is easy to check the rotational invariance of those quantities. So the PDE system (1) is rotationally invariant. Furthermore, for F_u and F_v to be shift invariant, the control functions a_j and b_j must be independent of (x,y). This is summarized in the following proposition and the proof is presented in Appendix A.

Proposition 2.1. *Suppose the PDE system (1) is translationally invariant, then the control functions $\{a_j\}_{j=0}^{16}$ and $\{b_j\}_{j=0}^{16}$ must be independent of (x,y).*

So the simplest choice of F_u and F_v is a linear combination of these differential invariants, leading to the following forms:

$$\begin{aligned} F_u(u,v,\mathbf{a}) &= \sum_{j=0}^{16} a_j(t)\mathrm{inv}_j(u,v),\\ F_v(v,u,\mathbf{b}) &= \sum_{j=0}^{16} b_j(t)\mathrm{inv}_j(v,u). \end{aligned} \tag{3}$$

Table 2
Translationally and rotationally invariant fundamental differential invariants up to the second order.

j	$\mathrm{inv}_j(u,v)$
0,1,2	1, v, u
3,4	$\|\nabla v\|^2 = v_x^2+v_y^2$, $\|\nabla u\|^2 = u_x^2+u_y^2$
5	$(\nabla v)^T\nabla u = v_xu_x+v_yu_y$
6,7	$\mathrm{tr}(\mathbf{H}_v) = v_{xx}+v_{yy}$, $\mathrm{tr}(\mathbf{H}_u) = u_{xx}+u_{yy}$
8	$(\nabla v)^T\mathbf{H}_v\nabla v = v_x^2v_{xx}+2v_xv_yv_{xy}+v_y^2v_{yy}$
9	$(\nabla v)^T\mathbf{H}_u\nabla v = v_x^2u_{xx}+2v_xv_yu_{xy}+v_y^2u_{yy}$
10	$(\nabla v)^T\mathbf{H}_v\nabla u = v_xu_xv_{xx}+(v_xu_y+v_yu_x)v_{xy}+v_yu_yv_{yy}$
11	$(\nabla v)^T\mathbf{H}_u\nabla u = v_xu_xu_{xx}+(v_xu_y+v_yu_x)u_{xy}+v_yu_yu_{yy}$
12	$(\nabla u)^T\mathbf{H}_v\nabla u = u_x^2v_{xx}+2u_xu_yv_{xy}+u_y^2v_{yy}$
13	$(\nabla u)^T\mathbf{H}_u\nabla u = u_x^2u_{xx}+2u_xu_yu_{xy}+u_y^2u_{yy}$
14	$\mathrm{tr}(\mathbf{H}_v^2) = v_{xx}^2+2v_{xy}^2+v_{yy}^2$
15	$\mathrm{tr}(\mathbf{H}_v\mathbf{H}_u) = v_{xx}u_{xx}+2v_{xy}u_{xy}+v_{yy}u_{yy}$
16	$\mathrm{tr}(\mathbf{H}_u^2) = u_{xx}^2+2u_{xy}^2+u_{yy}^2$

3. Training PDEs via an optimal control approach

In this section, we propose an optimal control framework[3] to train the intelligent PDE system for particular vision tasks.

3.1. The objective functional

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

$$\begin{aligned} J\left(\{u_m\}_{m=1}^M,\left\{a_j\right\}_{j=0}^{16},\left\{b_j\right\}_{j=0}^{16}\right) &= \frac{1}{2}\sum_{m=1}^{M}\int_\Omega[u_m(x,y,1)-O_m]^2\mathrm{d}\Omega\\ &+\frac{1}{2}\sum_{j=0}^{16}\lambda_j\int_0^1a_j^2(t)\mathrm{d}t+\frac{1}{2}\sum_{j=0}^{16}\mu_j\int_0^1b_j^2(t)\mathrm{d}t, \end{aligned} \tag{4}$$

where $u_m(x,y,1)$ is the output image at time $t=1$ computed from Eq.(1) when the input image is I_m, and λ_i and μ_i are positive weighting parameters. The first term requires that the final output of our PDE system be close to the ground truth. The second and the third terms are for regularization so that the optimal control problem is well posed, as there may be multiple minimizers for the first term.

3.2. Solving the optimal control problem

Then we have the following optimal control problem with PDE constraints:

$$\min_{\left\{a_j\right\}_{j=0}^{16},\left\{b_j\right\}_{j=0}^{16}} J\left(\{u_m\}_{m=1}^M,\left\{a_j\right\}_{j=0}^{16},\left\{b_j\right\}_{j=0}^{16}\right), \tag{5}$$

$$s.t.\begin{cases} \dfrac{\partial u_m}{\partial t}-F_u\left(u_m,v_m,\left\{a_j\right\}_{j=0}^{16}\right)=0, & (x,y,t)\in Q,\\ u_m(x,y,t)=0, & (x,y,t)\in\Gamma,\\ u_m|_{t=0}=f_{u_m}, & (x,y)\in\Omega,\\ \dfrac{\partial v_m}{\partial t}-F_v\left(v_m,u_m,\left\{b_j\right\}_{j=0}^{16}\right)=0, & (x,y,t)\in Q,\\ v_m(x,y,t)=0, & (x,y,t)\in\Gamma,\\ v_m|_{t=0}=f_{v_m}, & (x,y)\in\Omega. \end{cases} \tag{6}$$

By introducing the adjoint equation of Eq. (6), the Gâteaux derivative (see Appendix B.1) of J can be computed and consequently, the (local) optimal $\{a_j\}_{j=0}^{16}$ and $\{b_j\}_{j=0}^{16}$ can be computed via gradient-based algorithms (e.g., conjugate gradient). Here we give the form of adjoint equations and Gâteaux derivative directly, while leaving the deduction details in Appendix C.

3.2.1. Adjoint equation

The adjoint equations of φ_m and ϕ_m can be written as:

$$\begin{cases} \dfrac{\partial\varphi_m}{\partial t}+E(u_m,v_m,\varphi_m,\phi_m)=0, & (x,y,t)\in Q,\\ \varphi_m=0, & (x,y,t)\in\Gamma,\\ \varphi_m|_{t=1}=O_m-u_m(1), & (x,y)\in\Omega,\\ \dfrac{\partial\phi_m}{\partial t}+\tilde{E}(v_m,u_m,\phi_m,\varphi_m)=0, & (x,y,t)\in Q,\\ \phi_m=0, & (x,y,t)\in\Gamma,\\ \phi_m|_{t=1}=0, & (x,y)\in\Omega, \end{cases} \tag{7}$$

[3] We sketch the existing theory of optimal control governed by PDEs that we will borrow in Appendix B.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

where

$$E(u_m,v_m,\varphi_m,\phi_m)=\sum_{(p,q)\in\mathcal{P}}(-1)^{p+q}\frac{\partial^{p+q}\left(\sigma_{pq}(u_m)\varphi_m+\sigma_{pq}(v_m)\phi_m\right)}{\partial x^p\partial y^q},$$

$$\tilde{E}(v_m,u_m,\phi_m,\varphi_m)=\sum_{(p,q)\in\mathcal{P}}(-1)^{p+q}\frac{\partial^{p+q}\left(\tilde{\sigma}_{pq}(u_m)\varphi_m+\tilde{\sigma}_{pq}(v_m)\phi_m\right)}{\partial x^p\partial y^q},$$

$$\sigma_{pq}(u)=\frac{\partial F_u}{\partial u_{pq}}=\sum_{j=0}^{16}a_j\frac{\partial \mathrm{inv}_j(u,v)}{\partial u_{pq}},\sigma_{pq}(v)=\frac{\partial F_v}{\partial u_{pq}}=\sum_{j=0}^{16}b_j\frac{\partial \mathrm{inv}_j(v,u)}{\partial u_{pq}},$$

$$\tilde{\sigma}_{pq}(u)=\frac{\partial F_u}{\partial v_{pq}}=\sum_{j=0}^{16}a_j\frac{\partial \mathrm{inv}_j(u,v)}{\partial v_{pq}},\tilde{\sigma}_{pq}(v)=\frac{\partial F_v}{\partial v_{pq}}=\sum_{j=0}^{16}b_j\frac{\partial \mathrm{inv}_j(v,u)}{\partial v_{pq}},$$

$$u_{pq}=\frac{\partial^{p+q}u}{\partial x^p\partial y^q},v_{pq}=\frac{\partial^{p+q}v}{\partial x^p\partial y^q}.$$

Note that the adjoint equations evolve backwards from $t=1$ to $t=0$.

3.2.2. Gâteaux derivative

With the help of the adjoint equations, at each iteration the derivatives of J with respect to $a_j(t)$ and $b_j(t)$ are as follows:

$$\begin{aligned}\frac{DJ}{Da_j}&=\lambda_j a_j-\sum_{m=1}^{M}\int_\Omega\varphi_m inv_j(u_m,v_m)\mathrm{d}\Omega,\quad j=0,\ldots,16,\\ \frac{DJ}{Db_j}&=\mu_j b_j-\sum_{m=1}^{M}\int_\Omega\phi_m inv_j(v_m,u_m)\mathrm{d}\Omega,\quad j=0,\ldots,16,\end{aligned}\tag{8}$$

where the adjoint functions φ_m and ϕ_m are the solutions to Eq. (7).

3.3. Implementation issues

3.3.1. Discretization

To solve the intelligent PDE system and the adjoint equations numerically, we design a finite difference scheme [13] for the PDEs. We discretize the PDEs, i.e. replace the derivatives $\frac{\partial f}{\partial t}$, $\frac{\partial f}{\partial x}$ and $\frac{\partial^2 f}{\partial x^2}$ with finite differences as follows:

$$\begin{cases}\frac{\partial f}{\partial t}=\frac{f(t+\Delta t)-f(t)}{\Delta t},\\ \frac{\partial f}{\partial x}=\frac{f(x+1)-f(x)}{2},\\ \frac{\partial^2 f}{\partial x^2}=f(x-1)-2f(x)+f(x+1).\end{cases}\tag{9}$$

The discrete forms of $\frac{\partial f}{\partial y}$, $\frac{\partial^2 f}{\partial y^2}$ and $\frac{\partial^2 f}{\partial x\partial y}$ can be defined similarly. In addition, we discretize the integrations as

$$\begin{cases}\int_\Omega f(x,y)\mathrm{d}\Omega=\frac{1}{N}\sum_{(x,y)\in\Omega}f(x,y),\\ \int_0^t f(t)\mathrm{d}t=\Delta t\sum_{i=0}^{T_m}f(i\cdot\Delta t),\end{cases}\tag{10}$$

where N is the number of pixels in the spatial area, Δt is a properly chosen time step size and $T_m=\left\lfloor\frac{1}{\Delta t}+0.5\right\rfloor$ is the index of the expected output time. Then we use an explicit scheme to compute the numerical solutions of Eqs. (6) or (7).

3.3.2. Initialization

As the objective functional is non-convex, the convergence of the minimization process is toward a local minimum which depends on the initialization. In addition, a good initialization may save a lot of iterations. Hence the initialization issue is important for obtaining a good solution. In our current implementation, we set the initial functions of u and v as the input image:

$$u_m(x,y,0)=v_m(x,y,0)=I_m(x,y),m=1,2,\ldots,M.$$

Then we employ a heuristic method to initialize the control functions. We fix $b_j(t)=0$ and initialize $a_j(t)$ successively in time, where $j=0,1,\ldots,16$. At each time step, $\frac{\partial u_m}{\partial t}$ is expected to be $d_m(t)=\frac{O_m-u_m(t)}{1-t}$ so that $u_m(t)$ moves toward the expected output O_m. On the other hand, by the form of Eq. (1), we want $\{a_j(t)\}_{j=0}^{16}$ to minimize

$$\begin{aligned}&\sum_{m=1}^{M}\int_\Omega\left[F_u\left(u_m,v_m,\left\{a_j(t)\right\}_{j=0}^{16}\right)-d_m(t)\right]^2\mathrm{d}\Omega\\ &=\sum_{m=1}^{M}\int_\Omega\left[f_m^T(u_m,v_m)a(t)-d_m(t)\right]^2\mathrm{d}\Omega,\end{aligned}\tag{11}$$

where

$$\begin{aligned}\mathbf{f}_m(u_m,v_m)&=[\mathrm{inv}_0(u_m,v_m),\cdots,\mathrm{inv}_{16}(u_m,v_m)]^T,\\ \mathbf{a}(t)&=[a_0(t),\cdots,a_{16}(t)]^T.\end{aligned}$$

So $\mathbf{a}(t)$ is the solution to

$$\mathbf{F}\mathbf{a}(t)=\mathbf{d},\tag{12}$$

where

$$\begin{aligned}\mathbf{F}&=\sum_{m=1}^{M}\int_\Omega\mathbf{f}_m(u_m,v_m)\mathbf{f}_m^T(u_m,v_m,t)\mathrm{d}\Omega,\\ \mathbf{d}&=\sum_{m=1}^{M}\int_\Omega\mathbf{f}_m(u_m,v_m)d_m(t)\,\mathrm{d}\Omega.\end{aligned}$$

3.4. The optimal-control-based training framework

We now summarize in Algorithm 1 the optimal control training framework for the intelligent PDE system. After the PDE system is learnt, it can be applied to new test images by solving (1), whose inputs f_u and f_v are both the test images and the solution $u(t)|_{t=1}$ is the desired output image.

4. Extensions

The PDE formulation (1) is for grayscale images only. In practice, the images in vision tasks are often vector-valued, e.g., color images and multi-resolution images. To handle vector-valued images, this section extends the basic PDE formulation (1) to multiple channels, leading to a more general framework.

Algorithm 1. Optimal control framework for training the PDE system

Input: Training image pairs $\{(I_m,O_m)\}_{m=1}^{M}$.

1: Initialize $a_j(t)$, $t=0,\Delta t,\ldots,1-\Delta t$, by minimizing Eq. (12) and fix $b_j(t)=0, j=0,1,\ldots,16$. Let $u_m(x,y,0)=v_m(x,y,0)=I_m(x,y)$, $m=1,2,\ldots,M$.
2: **while** not converged **do**
3: Solve PDEs (6) for u_m and v_m, $m=1,\ldots,M$.
4: Solve PDEs (7) for φ_m and ϕ_m, $m=1,\ldots,M$.
5: Compute $\frac{DJ}{Da_j}$ and $\frac{DJ}{Db_j}$, $j=0,\ldots,16$, using Eq. (8).

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

6: Decide the search direction using the conjugate gradient method [14].
7: Perform golden search along the search direction and update $a_j(t)$ and $b_j(t)$, $j=0, \ldots, 16$.
8: **end while**

Output: The coefficient functions $\{a_j(t)\}_{j=0}^{16}$ and $\{b_j(t)\}_{j=0}^{16}$.

4.1. Handling vector-valued images

For color images, the design of PDEs becomes much more intricate because the correlation among different channels should be carefully handled so that spurious colors do not appear. Without sufficient intuitions on the correlation among different channels of images, people either consider a color image as a set of three images and apply PDEs independently [15], or use LUV color space instead, or from some geometric considerations [16]. For some vision problems such as denoising and inpainting, the above-mentioned methods may be effective. However, for more complex problems, such as Color2Gray [17], i.e., keeping the contrast among nearby regions when converting color images to grayscale ones, and demosaicing, i.e., inferring the missing colors from Bayer raw data [18], these methods may be incapable as human intuition may fail to apply. Consequently, we are also unaware of any PDE related work for these two problems.

By introducing fundamental differential invariants involving all channels and the extra indicator function, the channels are naturally coupled and the correlation among the channels is implicitly encoded in the control parameters. Specifically, the modifications on the framework for vector-valued images include:

1. A single output channel u now becomes multiple channels: u_k, $k=1, 2, 3$.
2. There are more shift and rotationally invariant fundamental differential invariants. The set of such invariants up to second order is as follows:

$$\left\{1, f_r, (\nabla f_r)^T \nabla f_s, (\nabla f_r)^T \mathbf{H}_{f_m} \nabla f_s, \mathrm{tr}\left(\mathbf{H}_{f_r}\right), \mathrm{tr}\left(\mathbf{H}_{f_r}\mathbf{H}_{f_s}\right)\right\}$$

   where f_r, f_s and f_m could be the indicator function or either channel of the output image. Now there are 69 elements in the set.
3. The initial function for the indicator function is the luminance of the input image.
4. The objective functional is the sum of J's in Eq. (4) for every channel.

Note that for vector-valued images with more than three channels, we may simply increase the number of channels. It is also possible to use other color spaces. However, we deliberately stick to RGB color space in order to illustrate the power of our framework. We use the luminance of the input image as the initial function of the indicator function because luminance is the most informative component of a color image, in which most of the important structural information, such as edges, corners and junctions, is well kept.

4.2. Multi-layer PDE systems

Although we have shown that our PDE system is a good regressor for many vision tasks, it may not be able to approximate *all* vision mappings at a desired accuracy. To improve their approximation power, a straightforward way is to introduce higher order differential invariants or use more complex combinations, beyond current linear combination, of fundamental invariants. However, the form of PDEs will be too complex to compute and analyze. Moreover, numerical instability may also easily occur. For example, if we use third-order differential invariants the magnitude of some invariants could be very small because many derivatives are multiplied together.[4] A similar situation also happens if a bilinear combination of the invariants is used. So it is not advised to add more complexity to the form of PDEs.

Recently, deep architectures [19,20], which are composed of multiple levels of nonlinear operations, are proposed for learning highly varying functions in vision and other artificial intelligence tasks [21]. Inspired by their work, we introduce a multi-layer PDE system. The forms of PDEs of each layer are the same, i.e., linear combination of invariants up to second order. The only difference is in the values of the control parameters and the initial values of all the functions, including the indicator function. The multi-layer structure is learnt by adopting a greedy strategy. After the first layer is determined, we use the output of the previous layer as the input of the next layer. The expected output of the next layer is still the ground truth image. The optimal control parameters for the next layer are determined as usual. As the input of the next layer is expected to be closer to the ground truth image than the input of the previous layer, the approximation accuracy can be successively improved. If there is prior information, such a procedure could be slightly modified. For example, for image demosaicing, we know that the Bayer raw data should be kept in the output full-color image. So we should replace the corresponding part of the output images with the Bayer raw data before feeding them to the next layer. The number of layers is determined by the training process automatically, e.g., the layer construction stops when a new layer does not result in output images with smaller error from the ground truth images.

5. Discussions

In this subsection, we would like to discuss and highlight the relationship between our intelligent PDE system and some related works.

5.1. Connections to traditional PDEs

5.1.1. Connection to diffusion equations

The mechanism of diffusion has been widely applied to solve problems arising in computer vision and image processing. The general diffusion PDEs can be written as

$$\frac{\partial u}{\partial t} - \mathrm{div}(c(x,y,t)\nabla u) = 0 \tag{13}$$

where $c(x,y,t)$ is the diffusion conductance. If c is a constant, i.e., independent of (x,y,t), it leads to a linear diffusion equation with a homogeneous diffusivity, i.e., heat equation [22]. A simple improvement would be to change c with the location (x,y) in the image, thus converting the equation into a linear anisotropic diffusion equation with nonhomogeneous diffusivity, e.g., the work in [23]. If the function c is image dependent, then the diffusion Eq. (13) becomes a nonlinear diffusion equation, e.g., $c(\|\nabla u\|^2)$ in the Perona–Malik model [2].

Now we try to understand the above diffusion equations in our intelligent PDE system. For isotropic diffusion, it is easy to check that the heat equation is a special case of Eq. (1) with $F_u = \mathrm{inv}_7(u,v)$. For anisotropic diffusion, we have that $\mathrm{div}(c\nabla u) = \mathrm{tr}(c\mathbf{H}_u) + (\nabla u)^T \nabla c$, which can be considered as a combination of $\mathrm{tr}(\mathbf{H}_u)$ and ∇u with the pre-defined function c. In this view, we can understand the PDE system (1) as a more general nonlinear combination of $\mathrm{tr}(\mathbf{H}_u)$ and ∇u with various differentials and coefficients.

5.1.2. Connection to the Mumford–Shah model

Mumford and Shah [24] have proposed to obtain a segmented image u from f by minimizing the functional

$$E_f(u,K) = \beta \int_\Omega (u-f)^2 \mathrm{d}x + \int_{\Omega/K} \|\nabla u\|^2 \mathrm{d}x + \alpha |K|,$$

[4] We normalize the grayscales to when computing.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

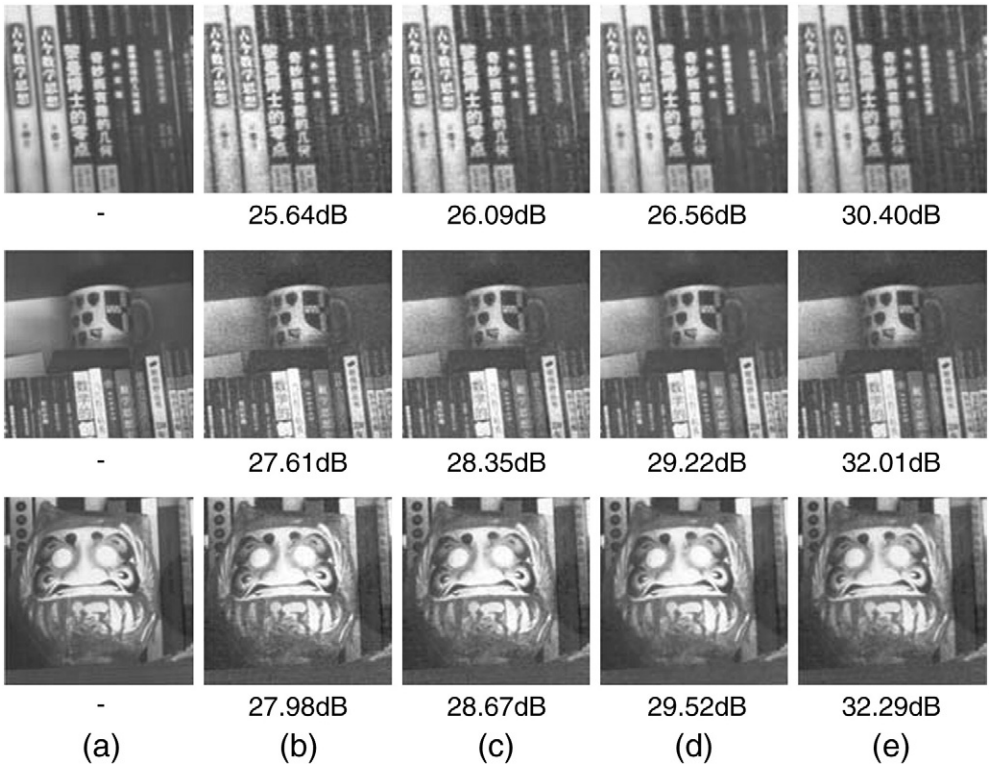

Fig. 1. The results of denoising images with natural noise. (a) Original noiseless image. (b) Noisy image with real noise. (c–e) Denoised images using ROF, TV-L^1, and our intelligent PDEs, respectively. The PSNRs are presented below each image.

where the discontinuity set K consists of the edges and its one-dimensional Hausdorff measure $|K|$ gives the total edge length. Based on the idea to approximate the discontinuity set K by a smooth function v, the Mumford–Shah functional can be approximated by the following functional [25]

$$F_f(u,v)=\int_\Omega\left(\beta(u-f)^2+v^2||\nabla u||^2+\alpha\left(d||\nabla v||^2+\frac{(1-v)^2}{4d}\right)\right)\mathrm{d}x,$$

where d is a positive parameter. Minimizing F_f corresponds to the following coupled evolutionary PDEs

$$\begin{cases}\dfrac{\partial u}{\partial t}-\mathrm{div}\left(v^2\nabla u\right)-\beta(f-u)=0,\\ \dfrac{\partial v}{\partial t}-d\Delta v+\dfrac{v}{\alpha}||\nabla u||^2-\dfrac{1-v}{4d}=0.\end{cases}\tag{14}$$

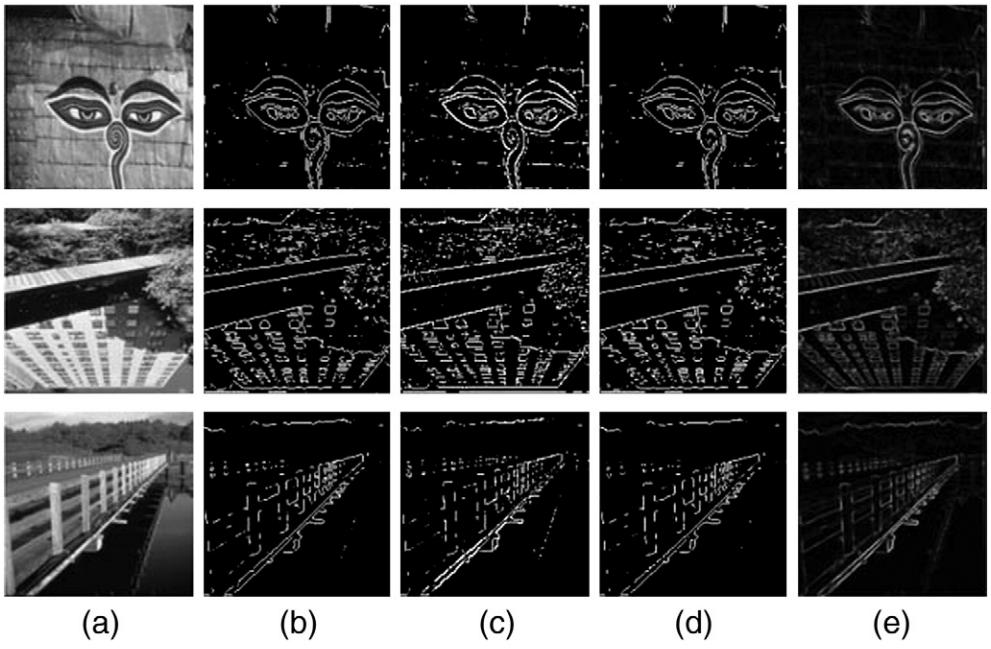

Fig. 2. The results of edge detection. (a) Original image. (b–e) Edge detection results using Sobel, Roberts Cross, Prewitt, and our intelligent PDE, respectively.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

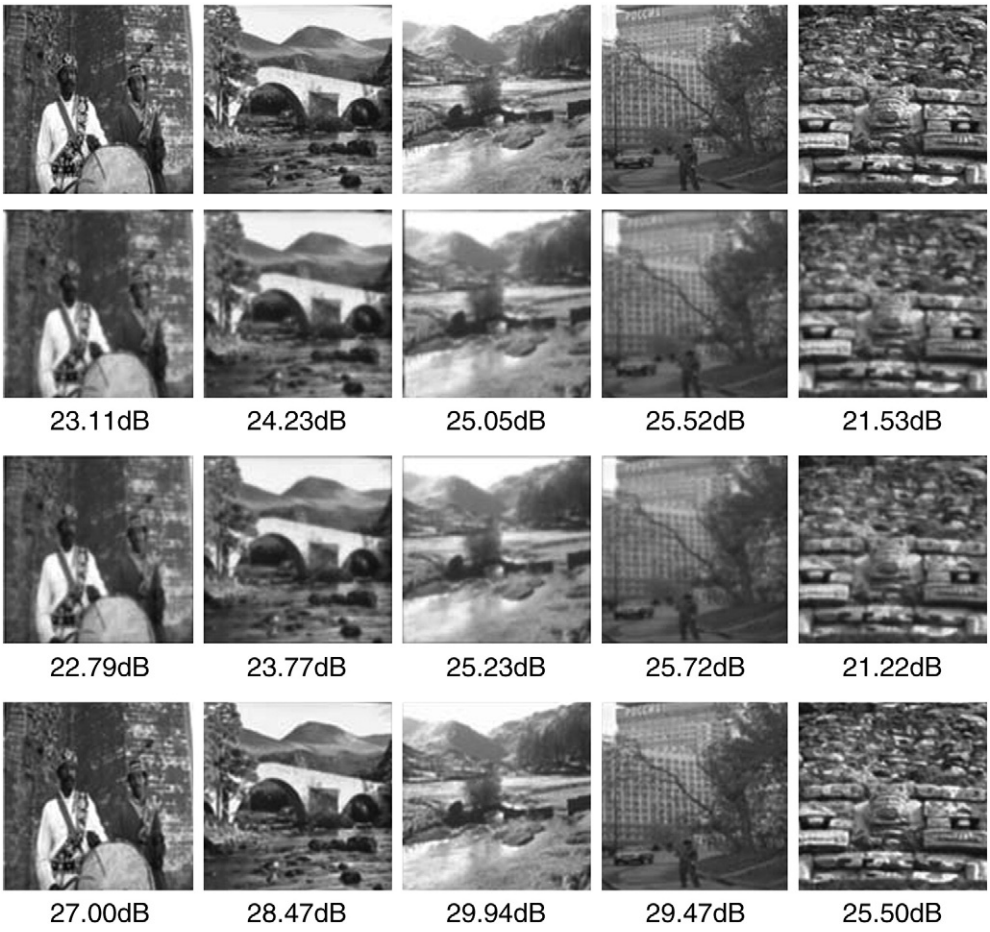

Fig. 3. The results of image blurring and deblurring. The top row are the original images. The second row are the blurring results of a Gaussian kernel. The third row are the blurring results of our intelligent PDEs. The bottom row are the deblurring results of our intelligent PDEs. The PSNRs are presented below each image.

In the view that $\mathrm{div}(v^2\nabla u)$ is also the combination of $\mathrm{tr}(\mathbf{H}_u)$ and ∇u with v, our intelligent PDE system can be considered as replacing v by its differentials invariants up to second order to generalize and improve the Mumford–Shah model.

5.2. Compassion to the work in Liu et al. [26]

The work in Liu et al. [26] also proposed a PDE-constrained optimal control framework for image restoration. But in that work, the authors only designed PDEs involving the curvature operator and differential invariants for basic image restoration tasks. In contrast, our work here proposes a more unified and elegant framework for more complex problems in computer vision.

The major differences between our intelligent PDE system and the PDEs in [26] are threefold: First, our intelligent PDE system is designed as a coupled evolutionary PDE with 34 differential invariants of the image u and the indicator v rather than a diffusion equation with 6 differential invariants of a single image u as in Liu et al. [26]. In this way, our framework forms a more general regressor to learn the mapping between the input and the output.[5] Furthermore, we proved the translationally and rotationally invariant properties for our intelligent PDE system, while the PDEs in Liu et al. [26] cannot be guaranteed due to the additional curvature operator in their PDE formulation.

Second, the PDEs in Liu et al. [26] was proposed by only considering the image restoration task, whereas our intelligent PDE system is designed for the general computer vision tasks, such as image denoising, edge detection, blurring and deblurring, image segmentation, etc.

Finally, the work in Liu et al. [26] can only handle image restoration problems with grayscale images. In contrast, our proposed framework can successfully solve various computer vision problems with both grayscale images and vector-valued images. Therefore, we obtained PDEs for some problems where traditional PDE methods have never been tried, such as Color2Gray and image demosaicing.

6. Experimental results

In this section, we apply our optimal control framework to learn PDEs for five groups of basic computer vision problems: natural image denoising, edge detection, blurring and deblurring, image segmentation, Color2Gray and image demosaicing. As our goal is to show that the data-based optimal control framework could be a new approach for designing PDEs and an effective regressor for many computer vision tasks, *not* to propose better algorithms for these tasks, we are not going to fine tune our PDEs and then compare it with the state-of-the-art algorithms in every task.

The experiments are categorized into the following five classes:

1. Test learning from ground truth by natural image denoising: $\{I_m\}$ are noisy images, $\{O_m\}$ are ground truth clean images.
2. Test learning from other methods by edge detection: $\{I_m\}$ are the original images, $\{O_m\}$ are results of other edge detectors.
3. Test learning to solve both primal and inverse problems by blurring and deblurring: $\{I_m\}$ are the original images, $\{O_m\}$ are blurred images for blurring and opposite for deblurring.

[5] Note that the differential invariants utilized in [26] is only a subset of Table 2.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

Fig. 4. 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.

4. Test learning from humans by image segmentation: $\{I_m\}$ are the original images, $\{O_m\}$ are manually segmented binary masks.
5. Test learning for vector-valued images: $\{I_m\}$ are the original images, $\{O_m\}$ are results of using the code provided by [17] for Color2Gray, and $\{I_m\}$ are the bilinearly interpolated color images, $\{O_m\}$ are the original full-color images for demosaicing.

6.1. Learning from ground truth: natural image denoising

Image denoising is one of the most fundamental low-level vision problems. For this task, we compare our learnt PDEs with the existing PDE-based denoising methods, ROF [5] and TV-L^1 [6], on images with unknown natural noise. This task is designed to demonstrate that our method can solve problems by learning from the ground truth. This is the first advantage of our data-based optimal control model. We take 240 images, each with a size of 150×150 pixels, of 11 objects using a Canon 30D digital camera, setting its ISO to 1600. For each object, 30 images are taken without changing the camera settings (by fixing the focus, aperture and exposure time) and without moving the camera position. The average image of them can be regraded as the noiseless ground truth image. We randomly choose 8 objects. For each object we randomly choose 5 noisy images. These noisy images and their ground truth images are used to train the PDE system. Then we compare our learnt PDEs with the traditional PDEs in [5] and TV-L^1 [6] on images of the remaining 3 objects.

Fig. 1 shows the comparison results. One can see that the PSNRs of our intelligent PDEs are dramatically higher than those of traditional PDEs. This is because our data-based PDE learning framework can easily adapt to unknown types of noise and obtain PDE forms to fit for the natural noise well, while most traditional PDE-based denoising methods were designed under specific assumptions on the types of noise (e.g., ROF is designed for Gaussian noise [5] while TV-L^1 is designed for impulsive noise [27]). Therefore, they may not fit for unknown types of noise as well as our intelligent PDEs.

6.2. Learning from other methods: Edge detection

The image edge detection task is used to demonstrate that our PDEs can be learnt from the results of different methods and achieve a better performance than all of them. This is another advantage of

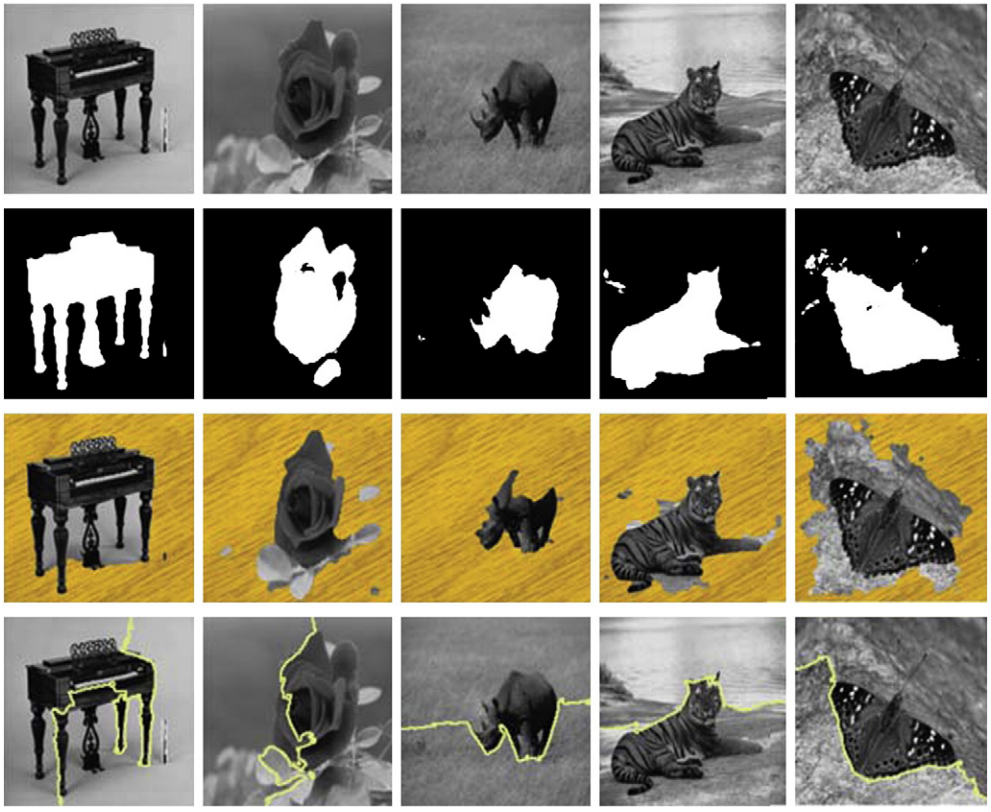

Fig. 5. The results of image segmentation. The top row are the input images. The second row are the masks obtained by thresholding the mask maps output by our learnt PDEs with a constant threshold 0.5. The third row are the segmentation results of active contour [31]. The bottom row are the results of normalized cut [32]. The results in the last two rows are produced using the original code by the authors.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

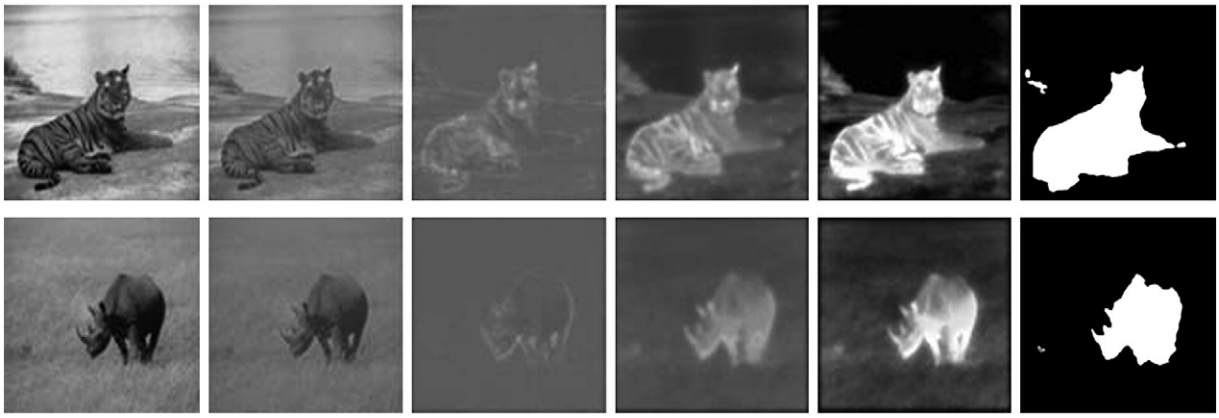

Fig. 6. The evolution of the mask maps. For each row, the first image is the input image, the second to the fifth are the mask maps at time $t = 0.25, 0.50, 0.75, 1.0$, respectively, and the last image is the final mask map with a threshold of 0.5.

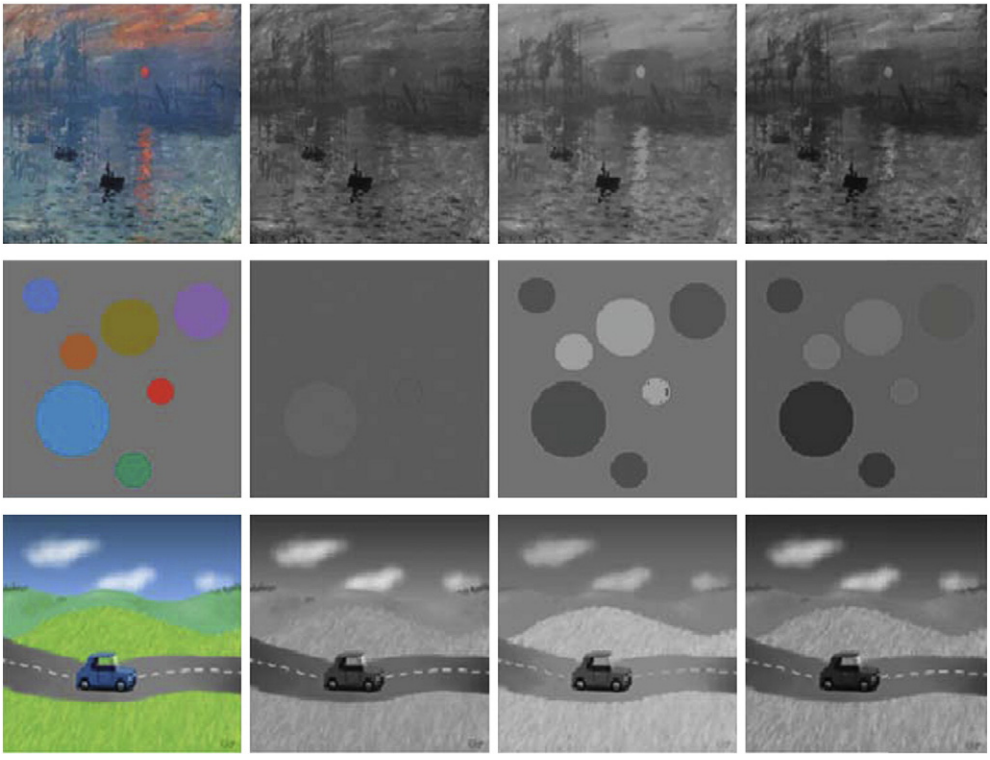

Fig. 7. Comparison of the results of Photoshop Grayscale (second column), Color2Gray [17] (third column) and our PDEs (fourth column). The first column are the original color images. These images are best viewed on screen. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

our data-based optimal control model. For this task, we use three simple first order edge detectors [28] (Sobel, Roberts Cross, and Prewitt) to generate the training data. We randomly choose 7 images from the Berkeley image database [29] and use the above three detectors to generate the output images,[6] together with the input images, to train our PDE system for edge detection.

Fig. 2 shows part of the edge detection results on other images in the Berkeley image database. One can see that our PDEs respond selectively to edges and basically produce visually significant edges, while the edge maps of other three detectors are more chaotic. Note that the solution to our PDEs is supposed to be a more or less smooth function. So one cannot expect that our PDEs produce an exactly binary edge map. Instead, an approximation of a binary edge map is produced.

[6] This implies that we actually use a kind of combination of the results from different methods to train our PDE system.

6.3. Learning to solve both primal and inverse problems: Blurring and deblurring

The traditional PDEs for solving different problems are usually of very different appearance. The task of solving both blurring and deblurring is designed to show that the same form of PDEs can be

Table 3
Comparison on PSNRs of different demosaicing algorithms. "BI + 1 layer" denotes the results of single-layer PDEs using bilinearly interpolated color images as the initial functions. Other abbreviations carry the similar meaning.

	BI	AP [33]	SA [18]	DFAPD [34]
Avg. PSNR (dB)	29.62 ± 2.91	37.83 ± 2.57	38.34 ± 2.56	38.44 ± 3.04
	BI + 1 layer	BI + 13 layers	DFAPD + 1 layer	DFAPD + 3 layers
Avg. PSNR (dB)	36.25 ± 2.32	37.80 ± 2.05	38.95 ± 2.93	38.99 ± 2.90

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

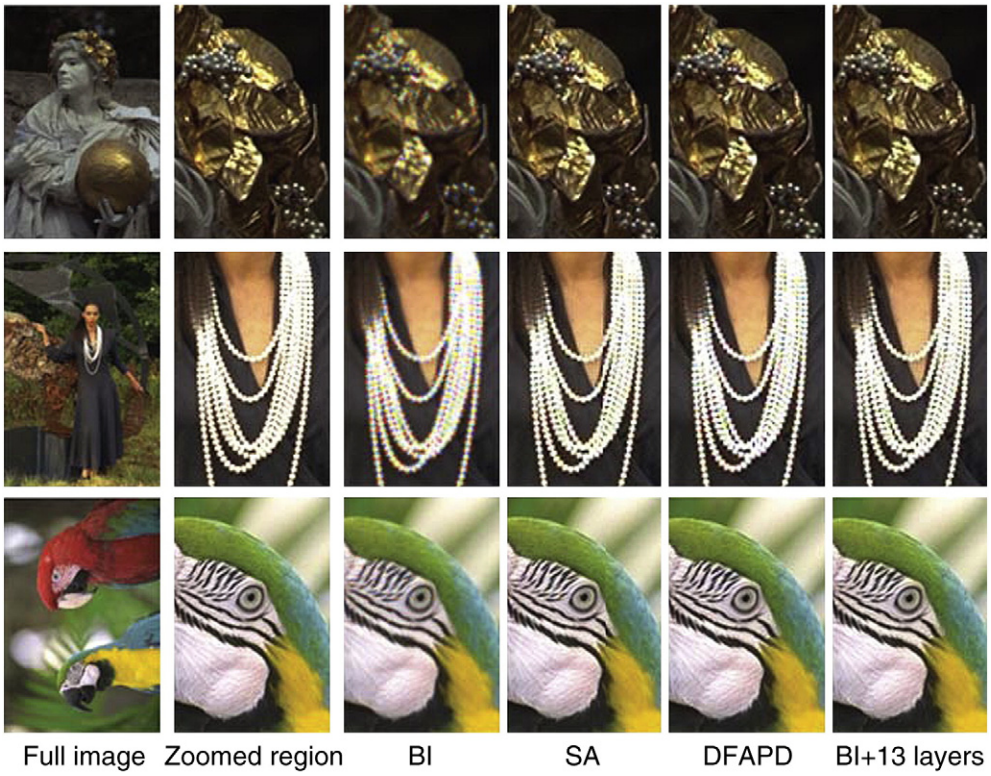

Fig. 8. Comparison of demosaicing results on Kodak images 17, 18 and 23. The first column are the full images. The second column are the zoomed-in regions in the full images. The third to sixth columns are the demosaicing results of different methods.

learnt to solve both the primal and inverse problems. This is the third advantage of our optimal control model.

For the image blurring task (the primal problem), the output image is the convolution of the input image with a Gaussian kernel. So we generate the output images by blurring high resolution images using a Gaussian kernel with $\sigma=1$. The original images are used as the input. As shown in the third row of Fig. 3, the output is nearly identical to the ground truth (the second row of Fig. 3). For the image deblurring task (the inverse problem), we just exchange the input and output images for training. One can see in the bottom row of Fig. 3 that the output is very close to the original image (first row of Fig. 3).

6.4. Learning from humans: Image segmentation

Image segmentation is designed to demonstrate that our PDE system can learn from the human behavior directly (learn the segmentation results provided by humans, e.g., manually segmented masks).

For image segmentation, it is a highly ill-posed problem and there are many criteria that define the goal of segmentation, e.g., breaking an image into regions with similar intensity, color, texture, or expected shape. As none of the current image segmentation algorithms can perform object level segmentation well out of complex backgrounds, 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 highly textured and simple thresholding cannot separate them. So we select 60 images from the Corel image database [30] that have relatively darker foregrounds and relatively simple backgrounds, but the foreground is not of uniformly lower gray levels than the background, and also prepare the manually segmented binary masks as the outputs of the training images, where the black regions are the backgrounds (Fig. 4).

Part of the segmentation results is shown in Fig. 5, where we have set a threshold for the output mask maps of our learnt PDEs with a constant 0.5. We see that our learnt PDEs produce fairly good object masks. We also test the active contour method by Li et al. [31][7] and the normalized cut method [32][8]. One can see from Fig. 5 that the active contour method cannot segment object details due to the smoothness constraint on the object shape and the normalized cut method cannot produce a closed foreground region. To provide a quantitative evaluation, we use the *F*-measure that merges the precision and recall of segmentation:

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

where

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

in which A is the ground truth mask and B is the computed mask. The most common choice of α is 2. On our test images, the F_2 measures of our PDEs, [31] and [32] 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.

We also present the evolution process of the mask maps across time (Fig. 6). One can see that although the foreground is relatively darker than the background, the PDEs correctly detect the most salient

[7] Code is available at http://www.engr.uconn.edu/~cmli/.
[8] Code is available at http://www.cis.upenn.edu/~jshi/software/.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

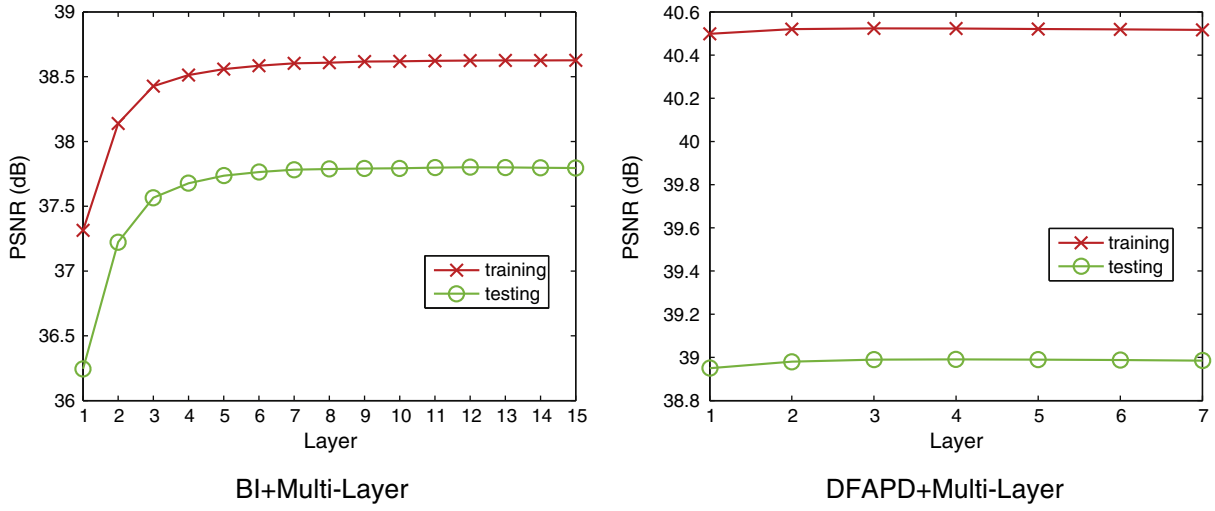

Fig. 9. Average PSNRs of each layer's output on training and testing images.

points/edges and then propagate the information across the foreground region, resulting in a *brighter* output region for the foreground.

6.5. Learning for vector-value images

In this subsection, we test our extended framework on two new vision applications, Color2Gray and demosaicing in which, to the best of our knowledge, PDEs have never been applied to before.

6.5.1. Color2Gray

We test our extended framework on the Color2Gray problem [17]. Contrast among nearby regions is often lost when color images are converted to grayscale by naively computing their luminance (e.g., using Adobe Photoshop Grayscale mode). Gooch et al. [17] proposed an algorithm to keep the contrast by attempting to preserve the salient features of the color image. Although the results are very impressive, the algorithm is very slow: $O(S^4)$ for an $S\times S$ square image. To learn the Color2Gray mapping, we choose 50 color images from the Corel database and generate their Color2Gray results using the code provided by Gooch et al. [17]. These 50 input–output image pairs are the training examples of our intelligent PDE system. We test the learnt PDEs on images in Gooch et al. [17] and their Web sites.[9] All training and testing images are resized to 150×150. Some results are shown in Fig. 7.[10] One can see (best viewed on screen) that our PDEs produce comparable visual effects to theirs. Note that the complexity of mapping with our PDEs is only $O(S^2)$: two orders faster than the original algorithm.

6.5.2. Image demosaicing

Commodity digital cameras use color filter arrays (CFAs) to capture raw data, which have only one channel value at each pixel. The missing channel values for each pixel have to be inferred in order to recover a full-color image. This technique is called demosaicing. The most commonly used CFAs are Bayer pattern [33,18,34]. Demosaicing is very intricate as many artifacts, such as blur, spurious color and zipper, may easily occur, and numerous demosaicing algorithms have been proposed (e.g., [33,18,34]). We show that with our intelligent PDE system, demosaicing also becomes easy.

We use the Kodak image database[11] for the experiment. Images 1–12 are used for training and images 13–24 are used for testing. To reduce the time and memory cost of training, we divide each 512×768 image to 12 non-overlapping 150×150 patches and select the first 50 patches with the richest texture, measured in their variances. Then we downsample the patches into Bayer CFA raw data, i.e., keeping only one channel value, indicated by the Bayer pattern, for each pixel. Finally, we bilinearly interpolate the CFA raw data (i.e., for each channel the missing values are bilinearly interpolated from their nearest four available values) into full-color images and use them as the input images of the training pairs. Note that bilinear interpolation is the most naive way of inferring the missing colors and many artifacts can occur. For comparison, we also provide results of several state-of-the-art algorithms [33,18,34] with the matlab codes provided by their authors. From Table 3, one can see that the results of multi-layer PDEs initialized with bilinear interpolation (BI) are comparable to state-of-the-art algorithms (also see Fig. 8). Our intelligent PDE system can also improve the existing demosaicing algorithms by using their output as our input images (see Table 3 for an example on DFAPD [34]). Fig. 9 shows the advantage of multi-layer PDEs over one-layer PDEs and the existence of an optimal layer number for both the training and the testing images.

7. Conclusion

In this paper, we have presented a framework for using data-based optimal control to learn PDEs as a general regressor to approximate the nonlinear mappings of different visual processing tasks. The experimental results on some computer vision and image processing problems show that our framework is promising. However, the current work is still preliminary. So we plan to improve and enrich our work in the following aspects. First, more theoretical issues should be addressed for this PDE system. For example, we will try to apply the Adomian decomposition method [35] to express the exact analytical solution to Eq. (1) and then analyze its physical properties. It is also attractive to provide deeper theoretical analysis for the indicator function v. Second, we would like to develop more computationally efficient numerical algorithms to solve our PDE-constrained optimal control problem (6). Third, we will apply our framework to more vision tasks to find out to what extent it works.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (Grant Nos. 61272341, 61231002, and 61173103), the National Natural Science Foundation of China-Guangdong Joint Fund (Grant No. U0935004), the Training Program of the Major Research

[9] http://www.cs.northwestern.edu/~ago820/color2gray/.

[10] Due to resizing, the results of Color2Gray in Fig. 7 are slightly different from those in Gooch et al. [17].

[11] The 24 images are available at http://www.site.uottawa.ca/~edubois/demosaicking.

Plan of the National Natural Science Foundation of China (Grant No. 91230103) and the Fundamental Research Funds for the Central Universities. The first author would also like to thank the support from China Scholarship Council.

Appendix A. Proof of Property 2.1

We prove that the coefficients $\{a_j\}_{j=0}^{16}$ and $\{b_j\}_{j=0}^{16}$ must be independent of (x,y).

Proof. We prove for $F_u(u,v,\{a_j\}_{j=0}^{16})$ in Eq. (1) only. We may rewrite

$$F_u\left(u,v,\left\{a_j\right\}_{j=0}^{16}\right)=\tilde{F}_u(u,v,x,y,t).$$

Then it suffices to prove that $\tilde{F}_u$ is independent of (x,y).

By the definition of translational invariance, when $I(x,y)$ changes to $I(x-x_0, y-y_0)$ by shifting with a displacement (x_0,y_0), $u(x,y)$ and $v(x,y)$ will change to $u(x-x_0, y-y_0)$ and $v(x-x_0, y-y_0)$, respectively. So the pair $u(x-x_0, y-y_0)$ and $v(x-x_0, y-y_0)$ fulfills Eq. (1), i.e.,

$$\frac{\partial u(x-x_0,y-y_0)}{\partial t}-\tilde{F}_u(u(x-x_0,y-y_0),v(x-x_0,y-y_0),x,y,t)=0.$$

Next, we replace $(x-x_0, y-y_0)$ in the above equation with (x,y) and have:

$$\frac{\partial u(x,y)}{\partial t}-\tilde{F}_u(u(x,y),v(x,y),x+x_0,y+y_0,t)=0.$$

On the other hand, the pair $(u(x,y), v(x,y))$ also fulfills Eq. (1), i.e.,

$$\frac{\partial u(x,y)}{\partial t}-\tilde{F}_u(u(x,y),v(x,y),x,y,t)=0.$$

Therefore, $\tilde{F}_u(u,v,x+x_0,y+y_0,t)=\tilde{F}_u(u,v,x,y,t)$, $\forall(x_0,y_0)$ that confines the input image inside Ω. So $\tilde{F}_u$ is independent of (x,y). □

Appendix B. PDE constrained optimal control

Now we sketch the existing theory of PDE constrained optimal control problem that we have used in Section 3. There are many types of these problems. Due to the scope of our paper, we only focus on the following optimal control problem:

$$\min J(f,g),\ s.t.\ \begin{cases}\dfrac{\partial f}{\partial t}-F(f,g)=0, & (x,t)\in Q,\\ f(x,t)=0, & (x,t)\in\Gamma,\\ f|_{t=0}=f_0, & x\in\Omega,\end{cases} \tag{B.1}$$

where J is a functional, $g\in\mathcal{U}$ controls f via the PDE, $\mathcal{U}$ is the admissible control set and $F(\cdot)$ is a smooth function.

Appendix B.1. Gâteaux derivative

Gâteaux derivative is an analogy and also an extension of the usual function derivative. Suppose $J(f)$ is a functional that maps a function f on region W to a real number. Its Gâteaux derivative (if it exists) is defined as the function f^* on W that satisfies:

$$\left(f^*,\delta f\right)_W=\lim_{\varepsilon\to 0}\frac{J(f+\varepsilon\cdot\delta f)-J(f)}{\varepsilon}$$

for all admissible perturbations δf of f. We may write f^* as $\frac{\mathrm{D}J}{\mathrm{D}f}$.

Appendix B.2. Solving problem (B.1) via Gâteaux derivative

We first introduce a Lagrangian function for problem (B.1):

$$\tilde{J}(f,g;\varphi)=J(f,g)+\int_\Omega\varphi\left[\frac{\partial f}{\partial t}-F(f,g)\right]\mathrm{d}\Omega, \tag{B.2}$$

where the multiplier φ can also be considered as adjoint function. It can be proved that the PDE constraint in Eq. (B.1) is exactly the first optimality condition of Eq. (B.2): $\frac{d\tilde{J}}{d\varphi}=0$, where $\frac{d\tilde{J}}{d\varphi}$ is the partial Gâteaux derivative of $\tilde{J}$ with respect to φ. For the second optimality condition $\frac{d\tilde{J}}{df}=0$, we call it the adjoint equation with respect to the PDE constraint in Eq. (B.1). One can also have that

$$\frac{\mathrm{D}J}{\mathrm{D}g}=\frac{\mathrm{d}\tilde{J}}{\mathrm{d}g}. \tag{B.3}$$

Thus $\frac{DJ}{Dg}=0$ is equivalent to the third optimality condition $\frac{d\tilde{J}}{d\varphi}=0$. Therefore, the problem (B.1) can be numerically solved if we can find the Gâteaux derivative of J with respect to the control g: we may find the optimal control g via gradient-based algorithms (e.g., conjugate gradient).

The above theory can be easily extended to systems of PDEs and multiple control functions. For more details and a more mathematically rigorous exposition of the above materials, please refer to [9].

Appendix C. Gâteaux derivatives of problem (6)

Now we show how to deduce the Gâteaux derivatives of Eq. (6) with respect to $a_j(t)$ and $b_j(t)$ at each iteration via adjoint equations. The Lagrangian function of optimal control system (6) is:

$$\begin{aligned}&\tilde{J}\left(\{u_m\}_{m=1}^M,\left\{a_j\right\}_{j=0}^{16},\left\{b_j\right\}_{j=0}^{16};\{\varphi_m\}_{m=1}^M,\{\phi_m\}_{m=1}^M\right)\\&=J\left(\{u_m\}_{m=1}^M,\left\{a_j\right\}_{j=0}^{16},\left\{b_j\right\}_{j=0}^{16}\right)\\&\quad+\sum_{m=1}^M\int_Q\varphi_m\left[\frac{\partial u_m}{\partial t}-F_u\left(u_m,v_m,\left\{a_j\right\}_{j=0}^{16}\right)\right]\mathrm{d}Q\\&\quad+\sum_{m=1}^M\int_Q\phi_m\left[\frac{\partial v_m}{\partial t}-F_v\left(v_m,u_m,\left\{b_j\right\}_{j=0}^{16}\right)\right]\mathrm{d}Q,\end{aligned} \tag{C.1}$$

where φ_m and ϕ_m are the adjoint functions.

It is known from the basic theory of optimal control that we can find adjoint function φ_m by solving another PDE, which is called the adjoint equation of Eq. (B.1). To find this adjoint equation for φ_m, we perturb $F_u(u,v)$ and $F_v(v,u)$ with respect to u. The perturbations can be written as follows:

$$\begin{aligned}&F_u(u+\varepsilon\cdot\delta u,v)-F_u(u,v)\\&\quad=\varepsilon\cdot\left(\frac{\partial F_u}{\partial u}(\delta u)+\frac{\partial F_u}{\partial u_x}\frac{\partial(\delta u)}{\partial x}+\cdots+\frac{\partial F_u}{\partial u_{yy}}\frac{\partial^2(\delta u)}{\partial y^2}\right)+o(\varepsilon)\\&\quad=\varepsilon\sum_{(p,q)\in\mathcal{P}}\sigma_{pq}(u)\frac{\partial^{p+q}(\delta u)}{\partial x^p\partial y^q}+o(\varepsilon),\\&F_v(v,u+\varepsilon\cdot\delta u)-F_v(v,u)\\&\quad=\varepsilon\sum_{(p,q)\in\mathcal{P}}\sigma_{pq}(v)\frac{\partial^{p+q}(\delta u)}{\partial x^p\partial y^q}+o(\varepsilon),\end{aligned} \tag{C.2}$$

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

where

$$\sigma_{pq}(u)=\frac{\partial F_u}{\partial u_{pq}}=\sum_{j=0}^{16}a_j\frac{\partial \mathrm{inv}_j(u,v)}{\partial u_{pq}},u_{pq}=\frac{\partial^{p+q}u}{\partial x^p\partial y^q},$$
$$\sigma_{pq}(v)=\frac{\partial F_v}{\partial u_{pq}}=\sum_{j=0}^{16}b_j\frac{\partial \mathrm{inv}_j(v,u)}{\partial u_{pq}},v_{pq}=\frac{\partial^{p+q}v}{\partial x^p\partial y^q}. \tag{C.3}$$

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

$$\begin{aligned}\delta\tilde{J}_{u_m}&=\tilde{J}(\cdots,u_m+\varepsilon\cdot\delta u_m,\cdots)-\tilde{J}(\cdots,u_m,\cdots)\\&=\frac{1}{2}\int_\Omega((u_m+\varepsilon\cdot\delta u_m)(x,y,1)-O_m(x,y))^2d\Omega\\&\quad-\frac{1}{2}\int_\Omega(u_m(x,y,1)-O_m(x,y))^2d\Omega\\&\quad+\int_Q\varphi_m\{[(u_m+\varepsilon\cdot\delta u_m)_t-F_u(u_m+\varepsilon\cdot\delta u_m,v_m)]-[(u_m)_t-F_u(u_m,v_m)]\}\mathrm{d}Q\\&\quad+\int_Q\phi_m\{[(v_m)_t-F_v(v_m,u_m+\varepsilon\cdot\delta u_m)]-[(v_m)_t-F_v(v_m,u_m)]\}\mathrm{d}Q\\&=\varepsilon\int_\Omega(u_m(x,y,1)-O_m(x,y))\delta u_m(x,y,1)d\Omega+\varepsilon\int_Q\varphi_m(\delta u_m)_t\mathrm{d}Q\\&\quad-\varepsilon\int_Q\varphi_m\sum_{(p,q)\in\mathcal{P}}\sigma_{pq}(u_m)\frac{\partial^{p+q}(\delta u_m)}{\partial x^p\partial y^q}\mathrm{d}Q\\&\quad-\varepsilon\int_Q\phi_m\sum_{(p,q)\in\mathcal{P}}\sigma_{pq}(v_m)\frac{\partial^{p+q}(\delta u_m)}{\partial x^p\partial y^q}\mathrm{d}Q+o(\varepsilon).\end{aligned} \tag{C.4}$$

As the perturbation δu_m should satisfy that

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

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

$$\varphi_m|_\Gamma=0,$$

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

$$\begin{aligned}\delta\tilde{J}_{u_m}&=\varepsilon\int_\Omega(u_m(x,y,1)-O_m(x,y))\delta u_m(x,y,1)\mathrm{d}\Omega\\&\quad+\varepsilon\int_\Omega(\varphi_m\cdot\delta u_m)(x,y,1)\mathrm{d}\Omega-\varepsilon\int_Q(\varphi_m)_t\delta u_m\mathrm{d}Q\\&\quad-\varepsilon\int_Q\sum_{(p,q)\in\mathcal{P}}(-1)^{p+q}\frac{\partial^{p+q}\left(\sigma_{pq}(u_m)\varphi_m\right)}{\partial x^p\partial y^q}\delta u_m\mathrm{d}Q\\&\quad-\varepsilon\int_Q\sum_{(p,q)\in\mathcal{P}}(-1)^{p+q}\frac{\partial^{p+q}\left(\sigma_{pq}(v_m)\phi_m\right)}{\partial x^p\partial y^q}\delta u_m\mathrm{d}Q+o(\varepsilon)\\&=\varepsilon\int_Q\left[(\varphi_m+u_m(x,y,1)-O_m(x,y))\delta(t-1)\right.\\&\quad\left.-(\varphi_m)_t-(-1)^{p+q}\frac{\partial^{p+q}\left(\sigma_{pq}(u_m)\varphi_m+\sigma_{pq}(v_m)\phi_m\right)}{\partial x^p\partial y^q}\right]\delta u_m\mathrm{d}Q+o(\varepsilon).\end{aligned} \tag{C.5}$$

By letting $\varepsilon\to0$, we have that the adjoint equation for φ_m is

$$\begin{cases}\dfrac{\partial\varphi_m}{\partial t}+E(u_m,v_m,\varphi_m,\phi_m)=0, & (x,y,t)\in Q,\\ \varphi_m=0, & (x,y,t)\in\Gamma,\\ \varphi_m|_{t=1}=O_m-u_m(1), & (x,y)\in\Omega,\end{cases} \tag{C.6}$$

in order that $\dfrac{\mathrm{d}\tilde{J}}{\mathrm{d}\mu_m}=0$, where

$$E(u_m,v_m,\varphi_m,\phi_m)=\sum_{(p,q)\in\mathcal{P}}(-1)^{p+q}\frac{\partial^{p+q}\left(\sigma_{pq}(u_m)\varphi_m+\sigma_{pq}(v_m)\phi_m\right)}{\partial x^p\partial y^q}. \tag{C.7}$$

Similarly, the adjoint equation for ϕ_m is

$$\begin{cases}\dfrac{\partial\phi_m}{\partial t}+\tilde{E}(v_m,u_m,\phi_m,\varphi_m)=0, & (x,y,t)\in Q,\\ \phi_m=0, & (x,y,t)\in\Gamma,\\ \phi_m|_{t=1}=0, & (x,y)\in\Omega,\end{cases} \tag{C.8}$$

where

$$\tilde{E}(v_m,u_m,\phi_m,\varphi_m)=\sum_{(p,q)\in\mathcal{P}}(-1)^{p+q}\frac{\partial^{p+q}\left(\tilde{\sigma}_{pq}(u_m)\varphi_m+\tilde{\sigma}_{pq}(v_m)\phi_m\right)}{\partial x^p\partial y^q},$$
$$\tilde{\sigma}_{pq}(u)=\frac{\partial F_u}{\partial v_{pq}}=\sum_{j=0}^{16}a_j\frac{\partial \mathrm{inv}_j(u,v)}{\partial v_{pq}},\ \tilde{\sigma}_{pq}(v)=\frac{\partial F_v}{\partial v_{pq}}=\sum_{j=0}^{16}b_j\frac{\partial \mathrm{inv}_j(v,u)}{\partial v_{pq}}. \tag{C.9}$$

The difference in $\tilde{J}$ caused by perturbing a_j is

$$\begin{aligned}\delta\tilde{J}_{a_j}&=\tilde{J}\left(\ldots,a_j+\varepsilon\cdot\delta a_j,\ldots\right)-\tilde{J}\left(\ldots,a_j,\ldots\right)\\&=\frac{1}{2}\lambda_j\int_0^1\left(a_j+\varepsilon\cdot\delta a_j\right)^2(t)dt-\frac{1}{2}\lambda_j\int_0^1a_j^2(t)\mathrm{d}t\\&\quad-\sum_{m=1}^{M}\int_Q\varphi_m\left[\left(a_j+\varepsilon\cdot\delta a_j\right)(t)-a_j(t)\right]inv_j(u_m,v_m)\mathrm{d}Q\\&=\varepsilon\int_0^1\left(\lambda_ja_j\cdot\delta a_j\right)(t)\mathrm{d}t-\varepsilon\int_0^1\left[\sum_{m=1}^{M}\int_\Omega\varphi_minv_j(u_m,v_m)d\Omega\right]\delta a_j\mathrm{d}t.\end{aligned} \tag{C.10}$$

Thus we have

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}a_j}=\lambda_ja_j-\sum_{m=1}^{M}\int_\Omega\varphi_m\mathrm{inv}_j(u_m,v_m)\mathrm{d}\Omega,\quad j=0,\ldots,16. \tag{C.11}$$

Similarly, we can also have

$$\frac{\mathrm{d}\tilde{J}}{\mathrm{d}b_j}=\mu_jb_j-\sum_{m=1}^{M}\int_\Omega\phi_m\mathrm{inv}_j(v_m,u_m)\mathrm{d}\Omega,\quad j=0,\ldots,16. \tag{C.12}$$

So by Eq. (B.3), we have that Eqs. (C.11) and (C.12) are also the Gâteaux derivatives of J with respect to a_j and b_j, respectively.

References

[1] T. Chen, J. Shen, Image processing and analysis: variational, PDE, wavelet, and stochastic methods, SIAM Publisher, 2005.
[2] P. Pietro, M. Jitendra, Scale-space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1990) 629–639.
[3] S. Osher, L. Rudin, Feature-oriented image enhancement using shock filters, SIAM J. Numer. Anal. 27 (1990) 919–940.
[4] G. Sapiro, Geometric partial differential equations and image analysis, Cambridge University Press, 2001.
[5] L. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D 60 (1992) 259–268.
[6] T. Chan, S. Esedoglu, Aspects of total variation regularizaed L^1 function approximation, SIAM J. Appl. Math. 65 (2005) 1817–1837.
[7] D. Strong, T. Chan, Edge-preserving and scale-dependent properties of total variation regularization, Inverse Prob. 19 (2003) 165–187.
[8] P. Olver, Applications of Lie Groups to Differential Equations, Springer-Verlarg, 1993.
[9] J. Lions, Optimal Control Systems Governed by Partial Differential Equations, Springer-Verlag, 1971.
[10] D. Kirk, Optimal Control Theory: An Introduction, Prentice-Hall, 1971.
[11] A. Ababnah, B. Natarajan, Optimal control-based strategy for sensor deployment, IEEE Trans. Syst. Man Cybern. B Cybern. 41 (2011) 97–104.
[12] B.M. ter Haar Romeny (Ed.), Geometry-Driven Diffusion in Computer Vision, Kluwer Academic Publishers, 1994.
[13] A. Jain, Partial differential equations and finite-difference methods in image processing, part 1: image representation, J. Optim. Theory Appl. 23 (1977) 65–91.
[14] J. Stoer, R. Bulirsch, Introduction to numerical analysis, Springer-Verlarg, 1998.
[15] M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester, Image inpainting, in: SIGGRAPH, 2000.
[16] D. Tschumperlé, R. Deriche, Vector-valued image regularization with PDEs: a common framework for different applications, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 506–517.
[17] A. Gooch, S. Olsen, J. Tumblin, B. Gooch, Color2Gray: salience-preserving color removal, ACM Trans. Graph. 24 (2005) 634–639.
[18] X. Li, Demosaicing by successive approximations, IEEE Trans. Image Process. 14 (2005) 267–278.
[19] G. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (2006) 1527–1554.
[20] Y. Bengio, P. Lamblin, P. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: NIPS, 2006.
[21] Y. Bengio, Y. LeCun, Scaling learning algorithms towards AI, in: Large Scale Kernel Machines, MIT Press, 2007.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004

[22] J. Koenderink, The structure of images, Biol. Cybern. 50 (1984) 363–370.
[23] G. Kim, E. Xing, L. Fei-Fei, T. Kanade, Distributed cosegmentation via submodular optimization on anisotropic diffusion, in: ICCV, 2011.
[24] D. Mumford, J. Shah, Boundary detection by minimizing functionals, in: CVPR, 1985.
[25] L. Ambrosio, V. Tortorelli, Approximation of functionals depending on jumps by elliptic functionals via γ-convergence, Commun. Pure Appl. Math. 7 (1992) 105–123.
[26] R. Liu, Z. Lin, W. Zhang, Z. Su, Learning PDEs for image restoration via optimal control, in: ECCV, 2010.
[27] N. Mila, A variational approach to remove outliers and impulse noise, J. Math. Imag. Vision 20 (2004) 99–120.
[28] J. Parker, Algorithms for Image Processing and Computer Vision, John Wiley & Sons Inc., 1997.
[29] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, in: ICCV, 2001.
[30] Corel photo library, Corel corporation, Ottawa, Canada.
[31] C. Li, C. Xu, C. Gui, M. Fox, Level set evolution without re-initialization: a new variational formulation, in: CVPR, 2005.
[32] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 888–905.
[33] B.K. Gunturk, Y. Altunbask, R.M. Mersereau, Color plane interpolation using alternating projections, IEEE Trans. Image Process. 11 (2002) 997C1013.
[34] D. Menon, S. Andriani, G. Calvagno, Demosaicing with directional filtering and a posteriori decision, IEEE Trans. Image Process. 16 (2007) 132–141.
[35] A. Wazwaz, Partial Differential Equations and Solitary Waves Theory, Springer-Verlag, 2009.

Please cite this article as: R. Liu, et al., Toward designing intelligent PDEs for computer vision: An optimal control approach, Image Vis. Comput. (2012), http://dx.doi.org/10.1016/j.imavis.2012.09.004