Learning Partial Differential Equations for Computer Vision*

Zhouchen Lin[1†] Wei Zhang[2] Xiaoou Tang[2]
[1]Microsoft Research Asia, zhoulin@microsoft.com
[2]The Chinese University of Hong Kong, {zw007,xtang}@ie.cuhk.edu.hk

Abstract

Partial differential equations (PDEs) have been successful for solving many problems in computer vision. However, the existing PDEs are all crafted by people with skill, based on some limited and intuitive considerations. As a result, the designed PDEs may not be able to handle complex situations in real applications. Moreover, human intuition may not apply if the vision task is hard to describe, e.g., object detection. These two aspects limit the wider applications of PDEs. In this paper, we propose a framework for learning a system of PDEs from real data to accomplish a specific vision task. As the first study on this problem, we assume that the system consists of two PDEs. One controls the evolution of the output. The other is for an *indicator function* that helps collect global information. Both PDEs are coupled equations between the output image and the indicator function, up to their second order partial derivatives. The way they are coupled is suggested by the shift and the rotational invariance that the PDEs should hold. The coupling coefficients are learnt from real data via an optimal control technique. Our framework can be extended in multiple ways. The experimental results show that learning-based PDEs could be an effective regressor for handling many different vision tasks. It is particularly powerful for those tasks that are difficult to describe by intuition.

1 Introduction

The applications of partial differential equations (PDEs) to computer vision and image processing date back to the 1960s [9, 14]. However, this technique did not draw much attention until the introduction of the concept of scale space by Koenderink [16] and Witkin [29] in the 1980s. Perona and Malik's work on anisotropic diffusion [25] finally drew great interest on PDE based methods. Nowadays, PDEs have been successfully applied to many problems

*Microsoft technical report #MSR-TR-2008-189. Patent filed.
†Corresponding author.

in computer vision and image processing [27, 7, 26, 2, 6], e.g., denoising [25], enhancement [22], inpainting [5], segmentation [17], stereo and optical flow computation.

In general, there are two 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 [25], shock filter [22] and curve evolution based equations [27, 26, 2, 6]). 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 variation of the energy functional (e.g., chapter 9 of [27]). Both methods require good skills in choosing appropriate functions and predicting the final effect of composing these functions such that the obtained PDEs roughly meet the goals. In either way, people have to heavily rely on their intuition, e.g., smoothness of edge contour and surface shading, on the vision task. Such intuition should easily 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. As a result, 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. If people do not have enough intuition on a vision task, they may have difficulty in acquiring effective PDEs. For example, can we have a PDE (or a PDE system) for object detection (Figure 1) that locates the object region if the object is present and does not respond if the object is absent? We believe that this is a big challenge to human intuition because it is hard to describe an object class, which may have significant variation in shape, texture and pose. Although there has been much work on PDE-based image segmentation, e.g., [17], the basic philosophy is always to follow the strong edges of the image and also require the edge contour to be smooth. Without using additional information to judge the content, the artificial PDEs always output an "object region" for any non-constant image. In short, current PDE design methods greatly limit the applications of PDEs to wider and more complex scopes. This motivates us to explore whether we can acquire PDEs that are not artificial yet more powerful.

Figure 1: What is the PDE (or PDE system) that can detect the object of interest (e.g., plane in left image) and does not respond if the object is absent (right image)?

In this paper, we propose a general framework for learning PDEs to accomplish a specific vision task. The vision task will be exemplified by a number of input-output image pairs, rather than relying on any form of human intuition. The learning algorithm is based on the theory of optimal control governed by PDEs [19]. As a preliminary investigation, we assume that the system consists of two PDEs. One controls the evolution of the output. 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 [21]. Currently, we only consider the case that the PDEs are linear combinations of fundamental differential invariants up to second order. The coupling coefficients are learnt from real data via the optimal control technique [19]. Our framework can be extended in different ways.

To our best knowledge, the only work of applying optimal control to computer vision and image processing is by Kimia et al. [15] and Papadakis et al. [24, 23]. However, the work in [15] is for minimizing *known* energy functionals for various tasks, including shape evolution, morphology, optical flow and shape from shading, where the target functions fulfill *known* PDEs. The methodology in [24, 23] is similar. Its difference from [15] is that it involves a control function that appears as the source term of *known* PDEs. In all existing work [15, 24, 23], the output functions are those that are desired. While in this paper, our goal is to determine a PDE system which is *unknown* at the beginning and the coefficients of the PDEs are those that are desired.

In principle, our learning-based PDEs form a regressor that learns a high dimensional mapping function between the input and the output. Many learning/approximation methods, e.g., neural networks, can also fulfill this purpose. However, learning-based PDEs are

fundamentally different from those methods in that those methods learn *explicit* mapping functions f: $O = f(I)$, where I is the input and O is the output, while our PDEs learn *implicit* mapping functions ϕ: $\phi(I, O) = 0$. Given the input I, we have to solve for the output O. The *input dependent* weights for the outputs, due to the coupling between the output and the indicator function that evolves from the input, makes our learning-based PDEs more adaptive to tasks and also require much fewer training samples. For example, we only used 50-60 training image pairs for all our experiments. Such a number is clearly impossible for traditional methods, considering the high dimensionality of the images and the high nonlinearity of the mappings. Moreover, backed by the rich theories on PDEs, it is possible to better analyze some properties of interest of the learnt PDEs. For example, the theory of differential invariants plays the key role in suggesting the form of our PDEs.

The rest of this paper is organized as follows. In Section 2, we introduce the related optimal control theory. In Section 3, we present our basic framework of learning-based PDEs. In Section 4, we testify to the effectiveness and versatility of our learning-based PDEs by five vision tasks. In Section 5, we show two possible ways of extending our basic framework to handle more complicated vision problems. Then we conclude our paper in Section 6.

2 Optimal Control Governed by Evolutionary PDEs

In this section, we sketch the existing theory of optimal control governed by PDEs that we will borrow. There are many types of these problems. Due to the scope of our paper, we only focus on the following distributed optimal control problem:

$$\text{minimize } J(f, u), \text{ where } u \in \mathcal{U} \text{ controls } f \text{ via the following PDE:} \tag{1}$$

$$\begin{cases} f_t &= L(\langle u \rangle, \langle f \rangle), & (\mathbf{x}, t) \in Q, \\ f &= 0, & (\mathbf{x}, t) \in \Gamma, \\ f|_{t=0} &= f_0, & \mathbf{x} \in \Omega, \end{cases} \tag{2}$$

in which J is a functional, $\mathcal{U}$ is the admissible control set and $L(\cdot)$ is a smooth function. The meaning of the notations can be found in Table 1. To present the basic theory, some definitions are necessary.

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	$\partial\Omega \times (0, T)$
W	Ω, Q, Γ, or $(0, T)$	$(f, g)_W$	$\int_W fg\mathrm{d}W$
∇f	gradient of f	$\mathbf{H}_f$	Hessian of f
$\wp$	$\{\emptyset, x, y, xx, xy, yy, \cdots\}$	$\lvert p\rvert$, $p \in \wp \cup \{t\}$	the length of string p
$\frac{\partial^{\lvert p\rvert} f}{\partial p}$, $p \in \wp \cup \{t\}$	$f, \frac{\partial f}{\partial t}, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial x \partial y}, \cdots$, when $p = \emptyset, t, x, y, xx, xy, \cdots$		
f_p, $p \in \wp \cup \{t\}$	$\frac{\partial^{\lvert p\rvert} f}{\partial p}$	$\langle f \rangle$	$\{f_p \lvert p \in \wp\}$
$P[f]$	the action of differential operator P on function f, i.e., if $P = a_0 + a_{10}\frac{\partial}{\partial x} + a_{01}\frac{\partial}{\partial y} + a_{20}\frac{\partial^2}{\partial x^2} + a_{11}\frac{\partial^2}{\partial x \partial y} + \cdots$, then $P[f] = a_0 f + a_{10}\frac{\partial f}{\partial x} + a_{01}\frac{\partial f}{\partial y} + a_{20}\frac{\partial^2 f}{\partial x^2} + a_{11}\frac{\partial^2 f}{\partial x \partial y} + \cdots$		
$L_{\langle f \rangle}(\langle f \rangle, \cdots)$	the differential operator $\sum_{p \in \wp} \frac{\partial L}{\partial f_p}\frac{\partial^{\lvert p\rvert}}{\partial p}$ associated to function $L(\langle f \rangle, \cdots)$		

2.1 Gâteaux Derivative of a Functional

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:

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

for all admissible perturbations δf of f. We may write f^* as $\frac{\mathrm{D}J}{\mathrm{D}f}$. For example, if $W = Q$ and $J(f) = \frac{1}{2}\int_\Omega [f(\mathbf{x}, T) - \tilde{f}(\mathbf{x})]^2 \mathrm{d}\mathbf{x}$, then

$$\begin{aligned}
& J(f + \varepsilon \cdot \delta f) - J(f) \\
= \; & \frac{1}{2}\int_\Omega [f(\mathbf{x}, T) + \varepsilon \cdot \delta f(\mathbf{x}, T) - \tilde{f}(\mathbf{x})]^2 \mathrm{d}\Omega - \frac{1}{2}\int_\Omega [f(\mathbf{x}, T) - \tilde{f}(\mathbf{x})]^2 \mathrm{d}\Omega \\
= \; & \varepsilon \int_\Omega [f(\mathbf{x}, T) - \tilde{f}(\mathbf{x})]\delta f(\mathbf{x}, T)\mathrm{d}\Omega + o(\varepsilon) \\
= \; & \varepsilon \int_Q \{[f(\mathbf{x}, t) - \tilde{f}(\mathbf{x})]\delta(t - T)\}\delta f(\mathbf{x}, t)\mathrm{d}Q + o(\varepsilon),
\end{aligned}$$

where $\delta(\cdot)$ is the Dirac function[1]. Therefore,

$$\frac{\mathrm{D}J}{\mathrm{D}f} = [f(\mathbf{x}, t) - \tilde{f}(\mathbf{x})]\delta(t - T).$$

2.2 Adjoint Differential Operator

The adjoint operator P^* of a linear differential operator P acting on functions on W is one that satisfies:

$$(P^*[f], g)_W = (f, P[g])_W,$$

for all f and g that are zero on ∂W and are sufficiently smooth[2]. The adjoint operator can be found by integration by parts, i.e., using the Green's formula [8]. For example, the adjoint operator of $P = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial}{\partial x}$ is $P^* = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} - \frac{\partial}{\partial x}$ because by Green's formula,

$$\begin{aligned}
(f, L[g])_\Omega &= \int_\Omega f(g_{xx} + g_{yy} + g_x)\mathrm{d}\Omega \\
&= \int_\Omega g(f_{xx} + f_{yy} - f_x)\mathrm{d}\Omega \\
&\quad + \int_{\partial\Omega} [(fg_x + fg - f_x g)\cos(N, x) + (fg_y - f_y g)\cos(N, y)]\mathrm{d}S \\
&= \int_\Omega (f_{xx} + f_{yy} - f_x) g \mathrm{d}\Omega,
\end{aligned}$$

where N is the outward normal of Ω and we have used that f and g vanish on $\partial\Omega$.

2.3 Finding the Gâteaux Derivative via Adjoint Equation

Problem (1)-(2) can be solved if we can find the Gâteaux derivative of J with respect to the control u: we may find the optimal control u via steepest descent.

Suppose $J(f, u) = \int_Q g(\langle u \rangle, \langle f \rangle)\mathrm{d}Q$, where g is a smooth function. Then it can be proved that

$$\frac{\mathrm{D}J}{\mathrm{D}u} = L^*_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[\varphi] + g^*_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[1], \tag{3}$$

where $L^*_{\langle u \rangle}$ and $g^*_{\langle u \rangle}$ are the adjoint operators of $L_{\langle u \rangle}$ and $g_{\langle u \rangle}$ (see Table 1 for the notations),

[1]Please do not confuse with the perturbations of functions.

[2]We are not to introduce the related function spaces in order to make this paper more intuitive.

respectively, and the adjoint function φ is the solution to the following PDE:

$$\begin{cases} -\varphi_t - L^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[\varphi] &= g^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[1], \quad (\mathbf{x}, t) \in Q, \\ \varphi &= 0, \quad (\mathbf{x}, t) \in \Gamma, \\ \varphi|_{t=T} &= 0, \quad \mathbf{x} \in \Omega, \end{cases} \tag{4}$$

which is called the *adjoint* equation of (2) (note that the adjoint equation evolves *backward* in time!). The proof can be found in Appendix 1.

The adjoint operations above make the deduction of the Gâteaux derivative non-trivial. As an equivalence, a more intuitive way is to introduce a Lagrangian function:

$$\tilde{J}(f, u; \varphi) = J(f, u) + \int_Q \varphi[f_t - L(\langle u\rangle, \langle f\rangle)]\mathrm{d}Q, \tag{5}$$

where the multiplier φ is exactly the adjoint function. Then one can see that the PDE constraint (2) is exactly the first optimality condition: $\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\varphi} = 0$, where $\frac{\mathrm{d}\tilde{J}}{\mathrm{d}\varphi}$ is the *partial* Gâteaux derivative of $\tilde{J}$ with respect to φ,[3] and verify that the adjoint equation is exactly the second optimality condition: $\frac{\mathrm{d}\tilde{J}}{\mathrm{d}f} = 0$. One can also have that

$$\frac{\mathrm{D}J}{\mathrm{D}u} = \frac{\mathrm{d}\tilde{J}}{\mathrm{d}u}. \tag{6}$$

So $\frac{\mathrm{D}J}{\mathrm{D}u} = 0$ is equivalent to the third optimality condition: $\frac{\mathrm{d}\tilde{J}}{\mathrm{d}u} = 0$.

As a result, we can use the definition of Gâteaux derivative to perturb f and u in $\tilde{J}$ and utilize Green's formula to pass the derivatives on the perturbations δf and δu to other functions, in order to obtain the adjoint equation and $\frac{\mathrm{D}J}{\mathrm{D}u}$ (for an example, see Appendix 3 for our real computation).

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 [19].

3 Learning-Based PDEs

Now we present our framework of learning PDE systems from training images. As preliminary work, we assume that our PDE system consists of two PDEs. One is for the evolution

[3] Namely, assuming f, u and φ are independent functions.

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

i	$\text{inv}_i(\rho, O)$
0,1,2	1, ρ, O
3,4,5	$\|\|\nabla\rho\|\|^2 = \rho_x^2 + \rho_y^2$, $(\nabla\rho)^t\nabla O = \rho_x O_x + \rho_y O_y$, $\|\|\nabla O\|\|^2 = O_x^2 + O_y^2$
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	$\text{tr}(\mathbf{H}_O^2) = O_{xx}^2 + 2O_{xy}^2 + O_{yy}^2$

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. This idea is inspired by the edge indicator in [27] (page 193). So our PDE system can be written as:

$$\begin{cases} O_t &= L_O(\mathbf{a}, \langle O\rangle, \langle\rho\rangle), & (\mathbf{x}, t) \in Q, \\ O &= 0, & (\mathbf{x}, t) \in \Gamma, \\ O|_{t=0} &= O_0, & \mathbf{x} \in \Omega; \\ \rho_t &= L_\rho(\mathbf{b}, \langle\rho\rangle, \langle O\rangle), & (\mathbf{x}, t) \in Q, \\ \rho &= 0, & (\mathbf{x}, t) \in \Gamma, \\ \rho|_{t=0} &= \rho_0, & \mathbf{x} \in \Omega, \end{cases} \tag{7}$$

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 O_0 and ρ_0 are the initial functions of O and ρ, respectively. 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 $T = 1$. L_O and L_ρ are smooth functions. $\mathbf{a} = \{a_i\}$ 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 will be discussed below.

3.1 Forms of PDEs

The space of all PDEs is infinite dimensional. To find the right one, we start with the properties that our PDE system should have, in order to narrow down the search space. We notice that most vision tasks 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.

According to the differential invariants theory in [21], L_O and L_ρ must be functions of the fundamental differential invariants under the groups of translation 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 of the function. We list those up to second order in Table 2.[4] 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 those quantities. In the sequel, we shall use $\text{inv}_i(\rho, O), i = 0, 1, \cdots, 16$, 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 $\text{inv}_i(O, \rho)$.

On the other hand, 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 2.

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

$$\begin{aligned} L_O(\mathbf{a}, \langle O\rangle, \langle\rho\rangle) &= \sum_{j=0}^{16} a_j(t)\text{inv}_j(\rho, O), \\ L_\rho(\mathbf{b}, \langle\rho\rangle, \langle O\rangle) &= \sum_{j=0}^{16} b_j(t)\text{inv}_j(O, \rho). \end{aligned} \tag{8}$$

[4] We add the constant function "1" for convenience of the mathematical deductions in the sequel.

Note that the visual process may not obey PDEs in such a form. However, we are *NOT* to discover what the real process is. Rather, we treat the visual process as a black box and regard the PDEs as a *regressor*. We only care whether the final output of our PDE system, i.e., $O(\mathbf{x}, 1)$, can *approximate* that of 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 needs and we need not care whether the system obeys either function. 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 a difficulty in theoretical analysis if higher order PDEs are considered. Nonetheless, as L_O and L_ρ in (8) are actually highly nonlinear and hence the dynamics of (7) can be very complex, they are already complex enough to approximate many vision tasks in our experiments, as will be shown in Sections 4 and 5. So we choose to leave the involvement of higher order derivatives to future work.

3.2 Determining the Coefficients by Optimal Control Approach

Given the forms of PDEs shown in (8), 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, $m = 1, 2, \cdots, M$, and compute the coefficient functions that minimize the following functional:

$$\begin{aligned} & J\left(\{O_m\}_{m=1}^{M}, \{a_j\}_{j=0}^{16}, \{b_j\}_{j=0}^{16}\right) \\ = \; & \frac{1}{2}\sum_{m=1}^{M}\int_{\Omega}[O_m(\mathbf{x},1) - \tilde{O}_m(\mathbf{x})]^2 \mathrm{d}\Omega + \frac{1}{2}\sum_{j=0}^{16}\lambda_j \int_0^1 a_j^2(t)\mathrm{d}t + \frac{1}{2}\sum_{j=0}^{16}\mu_j \int_0^1 b_j^2(t)\mathrm{d}t, \end{aligned} \tag{9}$$

where $O_m(\mathbf{x}, 1)$ is the output image at time $t = 1$ computed from (7) when the input image is I_m, and λ_j and μ_j 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. The regularization is important, particularly when the training

samples are limited.

Then we may compute the Gâteaux derivative $\frac{\mathrm{D}J}{\mathrm{D}a_j}$ and $\frac{\mathrm{D}J}{\mathrm{D}b_j}$ of J with respect to a_j and b_j using the theory in Section 2. The expressions of the Gâteaux derivatives can be found in Appendix 3. Consequently, the optimal a_j and b_j can be computed by steepest descent.

3.3 Implementation Details

3.3.1 Initial functions of O and ρ

Good initialization increases 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:

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

However, this is not the only choice. If the user has some prior knowledge about the input/output mapping, s/he can choose other initial functions. Some examples of different choices of initial functions can be found in Section 5.

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

We initialize $\{a_i(t)\}$ successively in time while fixing $b_i(t) \equiv 0$, $i = 0, 1, \cdots, 16$. Suppose the time stepsize is Δt when solving (7) with finite difference. At the first time step, without any prior information, $O_m(\Delta t)$ is expected to be $\Delta t \tilde{O}_m + (1 - \Delta t) I_m$. So $\partial O_m / \partial t|_{t=\Delta t}$ is expected to be $\tilde{O}_m - I_m$ and we may solve $\{a_i(0)\}$ such that

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

is minimized, i.e., to minimize the difference between the left and the right hand sides of (7), where the integration here should be understood as summation. After solving $\{a_i(0)\}$, we can have $O_m(\Delta t)$ by solving (7) at $t = \Delta t$. Suppose at the $(k+1)$-th step, we have solved $O_m(k\Delta t)$, then we may expect that $O_m((k+1)\Delta t) = \frac{\Delta t}{1-k\Delta t} \tilde{O}_m + \frac{1-(k+1)\Delta t}{1-k\Delta t} O_m(k\Delta t)$ 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 $\{a_i(0)\}$.

3.3.3 Choice of Parameters

When solving (7) with finite difference, we use an explicit scheme to solve $O_m(k\Delta t)$ and $\rho_m(k\Delta t)$ successively. However, the explicit scheme is conditionally stable: if the temporal stepsize is too large, we cannot obtain solutions with controlled error. As we have not investigated this problem thoroughly, we simply borrow the stability condition for two-dimensional heat equations $f_t = \kappa(f_{xx} + f_{yy})$:

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

where Δx and Δy are the spatial stepsizes. 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:

$$\Delta t \leq \frac{1}{4\max(a_7, b_7)}. \tag{10}$$

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 (10), and compute $O_m(k \cdot \Delta t + i \cdot \delta t)$, $i = 1, 2, ..., K$, successively until we obtain $O_m((k+1)\Delta t)$. We have found that this method works for all our experiments.

And there are other parameters to choose. As it does not seem to have a systematic way to analyze their optimal values, we simply fix their values as: $M = 50 \sim 60$, $\Delta t = 0.05$ and $\lambda_i = \mu_i = 10^{-7}$, $i = 0, \cdots, 16$.

4 Experimental Results

In this section, we present the results on five different computer vision/image processing tasks: blur, edge detection, denoising, segmentation and object detection. As our goal is to show that PDEs could be an effective regressor for many vision tasks, *NOT* to propose better algorithms for these tasks, we are not going to compare with state-of-the-art algorithms in every task.

Figure 2: Partial results on image blurring. The top row are the input images. The middle row are the outputs of our learnt PDE. The bottom row are the groundtruth images obtained by blurring the input image with a Gaussian kernel. One can see that the output of our PDE is visually indistinguishable from the groundtruth.

For each task, 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 test images. Part of the results are shown in Figures 2-10, respectively. Note that we do *not* scale the range of pixel values of the output to be between 0 and 255. Rather, we *clip* the values to be between 0 and 255. Therefore, the reader can compare the strength of response across different images.

For the image blurring task (Figure 2), the output image is expected to be the convolution of the input image with a Gaussian kernel. It is well known [27] that this corresponds to evolving with the heat equation. This equation is almost exactly learnt using our method. So the output is nearly identical to the groundtruth.

For the image edge detection task (Figure 3), we want our learnt PDE system to approximate the Canny edge detector. One can see that our PDEs respond strongly at strong edges and weakly at weak edges. Note that the Canny detector outputs binary edge maps while our PDEs can only output smooth functions. So it is difficult to approximate the Canny detector at high precision.

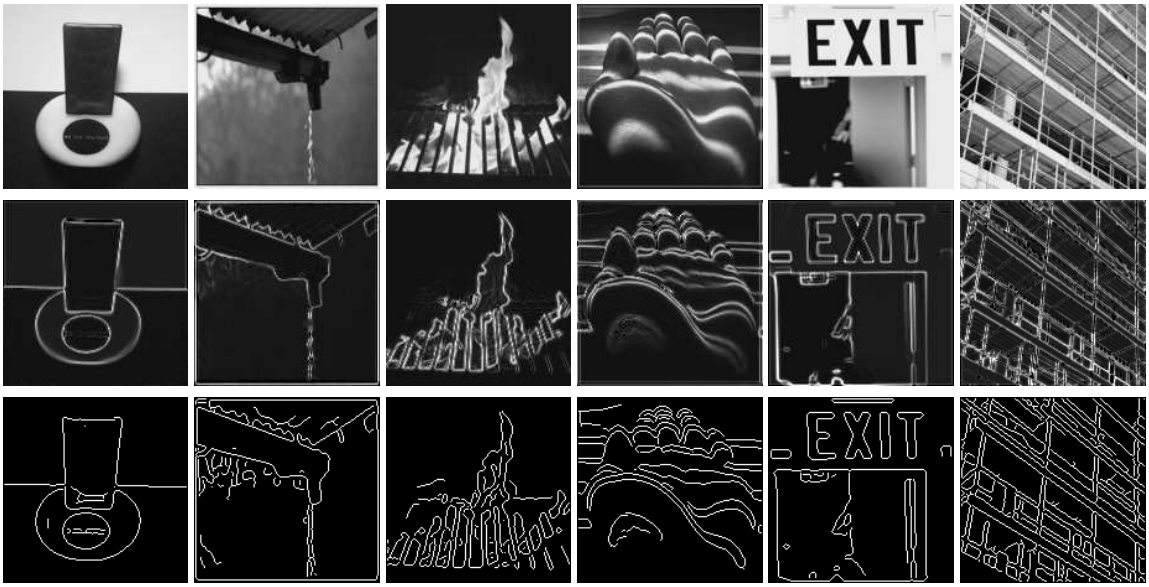


Figure 3: Partial results on edge detection. The top row are the input images. The middle row are the outputs of our learnt PDEs. The bottom row are the edge maps by the Canny detector. One can see that our PDEs respond strongly at strong edges and weakly at weak edges.

Figure 4: Partial results on image denoising. The top row are the input noisy images. The middle row are the outputs of our learnt PDEs. The bottom row are the outputs by the method in [10]. One can see that the results of our PDE system are comparable to those by [10] (using the original code by the authors).

For the image denoising task (Figure 4), we generate input images by adding Gaussian noise to the original images and use the original images as the ground truth. One can see that our PDEs suppress most of the noise while preserving the edges well. So we easily obtain PDEs that produce results comparable with those by [10] (using the original code by the authors), which was designed with a lot of wits.

While the above three vision problems can be solved by using local information only, we shall show that our learning-based PDE system is not simply a local regressor. Rather, it seems that it can automatically find the features of the problem to work with. Such a global effect may result from the indicator function that is supposed to collect the global information. We testify this observation by two vision tasks: image segmentation and object detection.

For the image segmentation task, we choose images with the foreground relatively darker than the background (but the foreground is *not* completely darker than the background so that a simple threshold can well separate them, see the second row of Figure 6) and prepare the manually segmented masks as the outputs of the training images (first row of Figure 5), where the black regions are the background. The segmentation results are shown in (Figure 6), 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. This may be because our PDEs correctly identify that graylevel is the key feature. We also test the active contour method by Li et al. [17] (using the original code by the authors). As it is designed to follow the strong edges and to prefer smooth object boundaries, its performance is not as good as ours.

For the object detection task, we require that the PDEs respond strongly inside the object region while they do not respond (or respond much more weakly) outside the object region. If the object is absent in the image, the response across the whole image should all be weak (Figure 1). It should be challenging enough for one to manually design such PDEs. As a result, we are unaware of any PDE-based method that can accomplish this task. The existing PDE-based segmentation algorithms always output an "object region" even if the image does

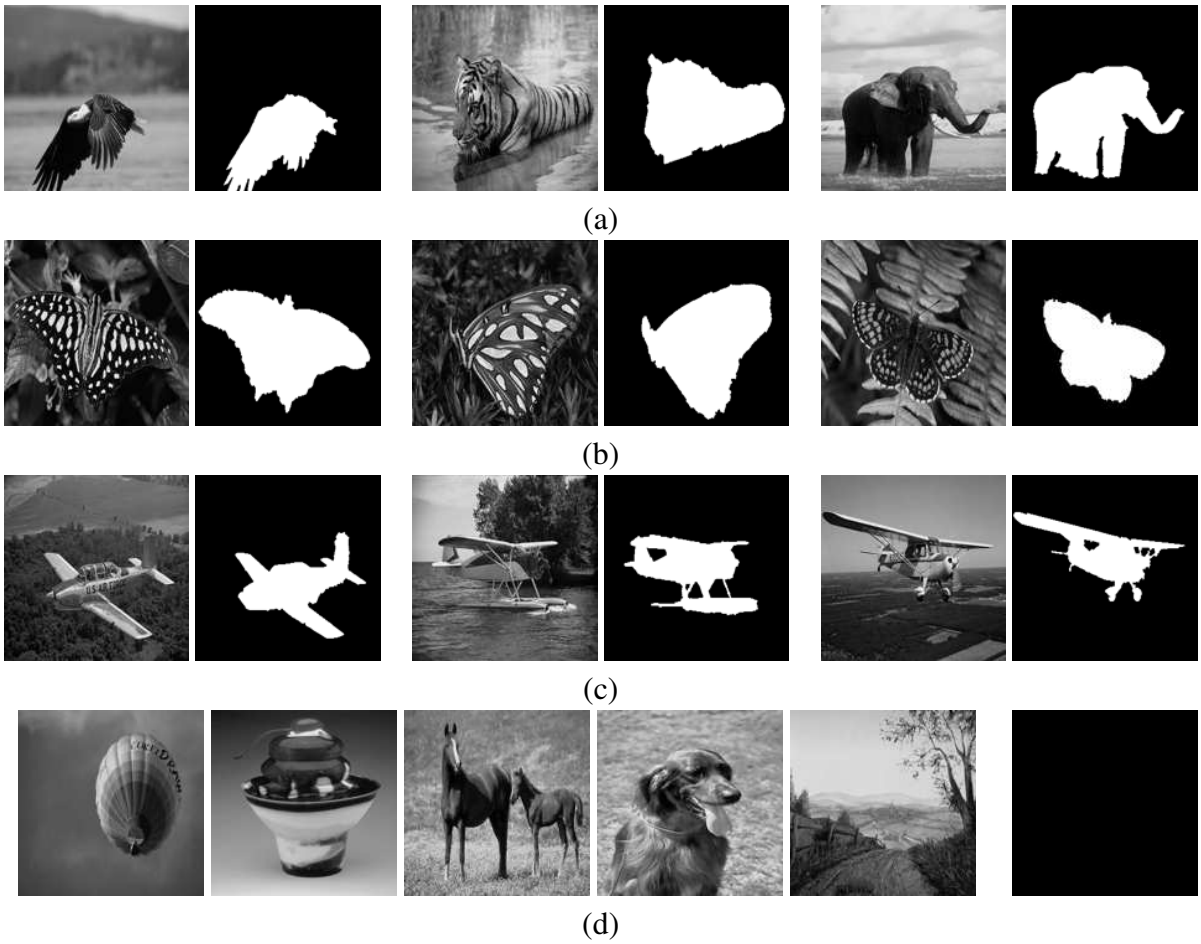


Figure 5: Examples of the training images for image segmentation (top row), butterfly detection (second and fourth rows) and plane detection (third and fourth rows). In each group of images of (a)∼(c), on the left is the input image and on the right is the groundtruth output mask. In (d), the left images are part of the input images as *negative samples* for training butterfly and plane detectors, where their groundtruth output masks are *all-zero* images (right).

Figure 6: Partial results of graylevel-based segmentation. The top row are the input images. The second row are the masks of foreground objects by thresholding with *image-dependent* thresholds. The third row are the mask obtained by thresholding the mask maps output by our learnt PDEs with a *constant* threshold 0.5. The bottom row are the segmentation results of [17] (best viewed on screen).

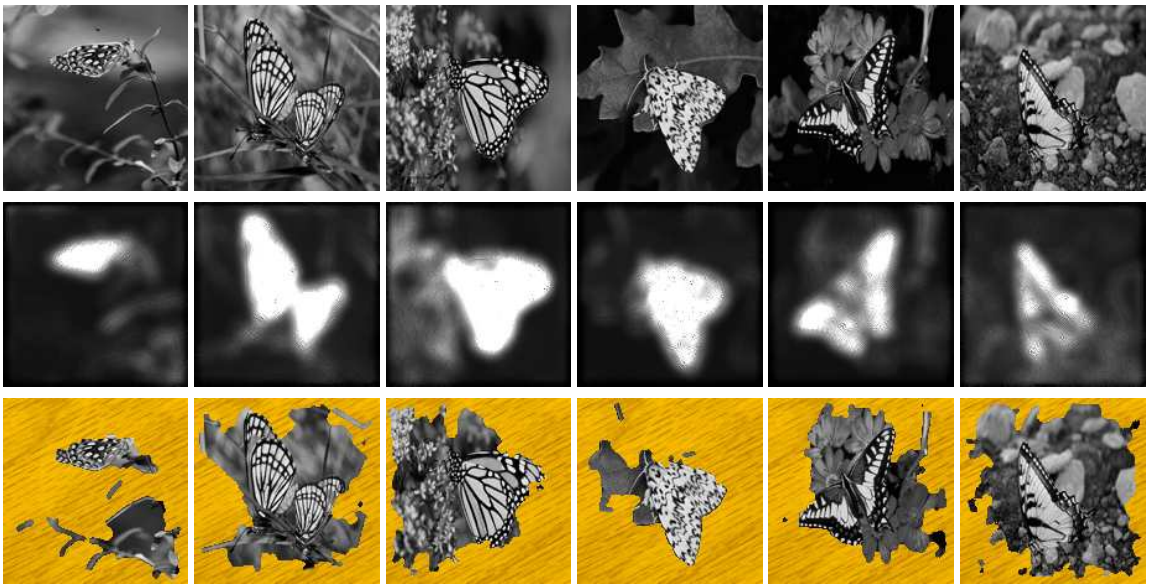


Figure 7: Partial results of detecting butterflies on images containing butterflies. The top row are the input images. The middle row are the output mask maps of our learnt PDEs. The bottom row are the segmentation results of [17] (best viewed on screen).

not contain the object of interest. In contrast, we will show that as desired the response of our learnt PDEs can be selective. In order to testify that our PDEs are able to identify different features, for this task we choose two data sets from Corel [1]: butterfly and plane. We select 50 images from each data set as positive samples and also prepare 10 images without the object of interest as negative samples (second to fourth rows of Figure 5). We also provide their groundtruth object masks in order to complete the training data.

The background and foreground of the "butterfly" and "plane" data sets (first rows of Figures 7 and 9) are very complex, so object detection is very difficult. One can see that our learnt PDEs respond strongly (the brighter, the stronger) in the regions of objects of interest, while the response in the background is relatively weak[5] (the second rows of Figures 7 and 9), even if the background also contains strong edges or rich textures, or has high graylevels. In contrast, artificial PDEs [17] mainly output the rich texture areas. We also apply the learnt object-oriented PDEs to images of other objects (the second rows of Figures 8 and 10). One can see that the response of our learnt PDEs is relatively low across the *whole*

[5]Note that as our learnt PDEs only *approximate* the desired vision task, one cannot expect that the outputs are exactly binary.

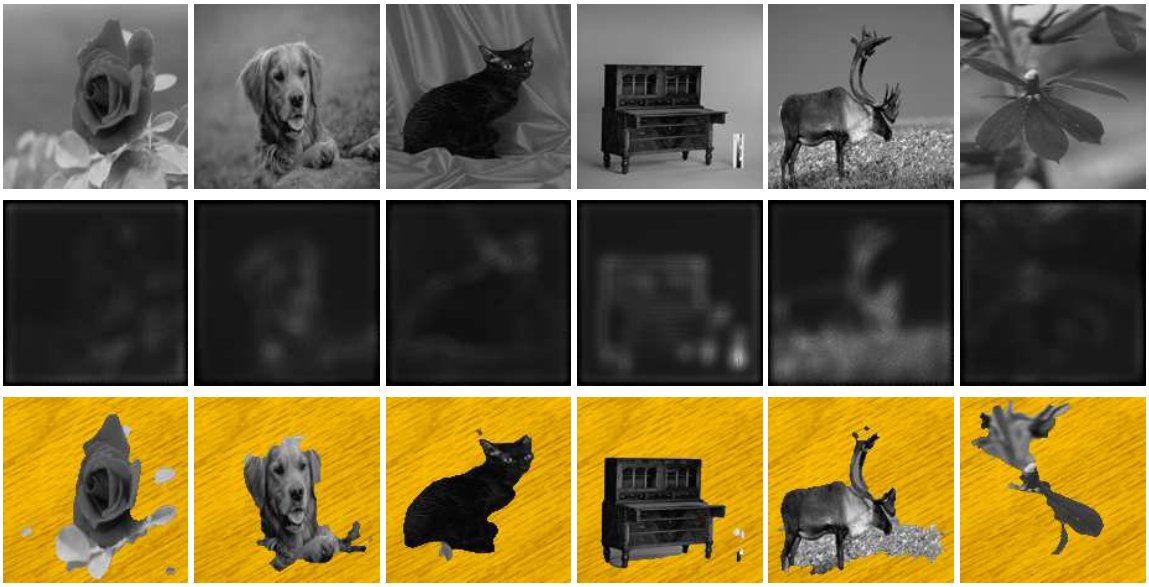


Figure 8: Partial results of detecting butterflies on images without butterflies. The top row are the input images. The middle row are the output mask maps of our learnt PDEs. The bottom row are the segmentation results of [17] (best viewed on screen). Please be reminded that the responses may appear stronger than they really are, due to the contrast with the dark background.

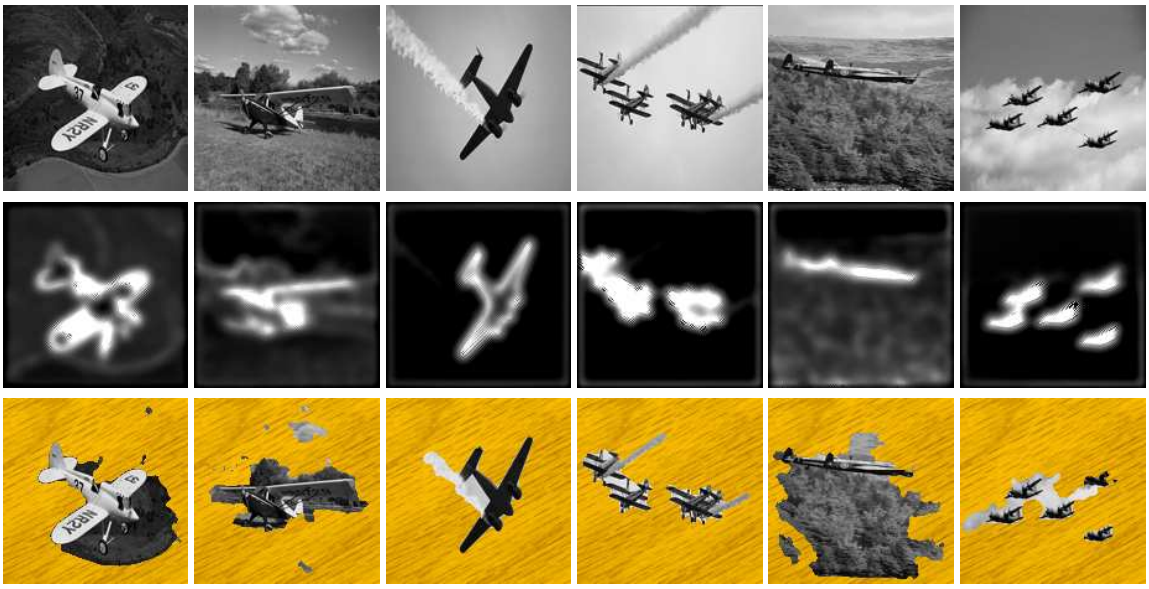


Figure 9: Partial results of detecting planes on images containing planes. The top row are the input images. The middle row are the output mask maps of our learnt PDEs. The bottom row are the segmentation results of [17] (best viewed on screen).

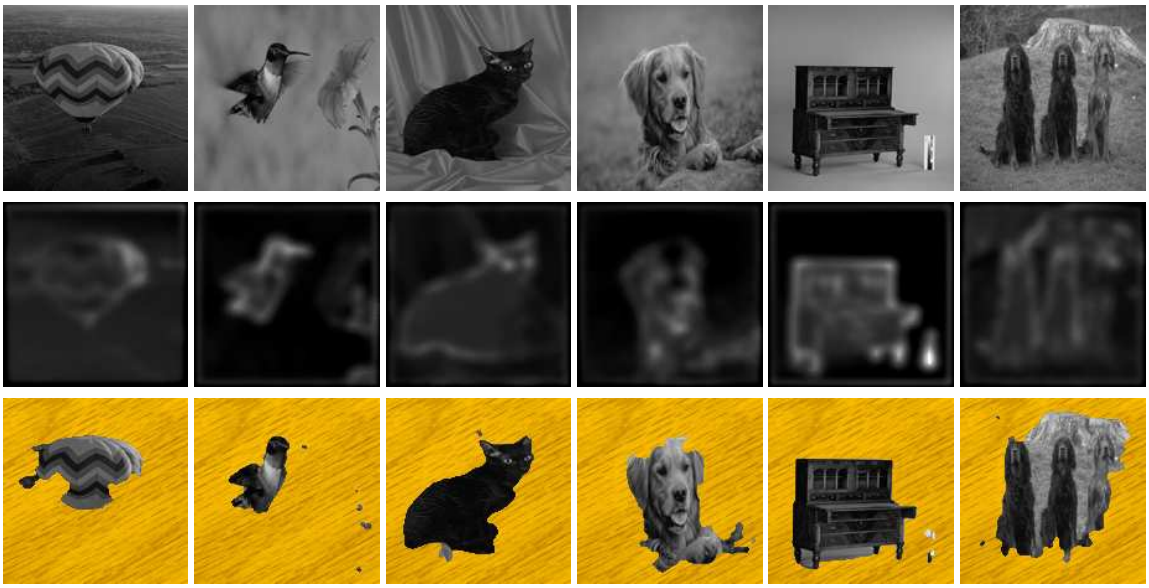


Figure 10: Partial results of detecting planes on images without planes. The top row are the input images. The middle row are the output mask maps of our learnt PDEs. The bottom row are the segmentation results of [17] (best viewed on screen). Please be reminded that the responses may appear stronger than they really are, due to the contrast with the dark background.

image[6]. In comparison, the method in [17] still outputs the rich texture regions. It seems that our PDEs automatically identify that the black-white patterns on the wings are the key feature of butterflies and the concurrent high-contrast edges/junctions/corners are the key feature of planes. The above examples show that our learnt PDEs are able to differentiate the object/non-object regions, without requiring the user to teach them what features are and what factors to consider.

5 Extensions

The framework presented in Section 3 is in the simplest formulation. It can be extended in multiple ways to handle more complex problems. Since we could not exhaust all the possibilities, we only present two examples.

[6]As clarified at the beginning of this Section, we present the output images by clipping values, not scaling values, to [0, 255]. So we can compare the strength of response in different images.

5.1 Handling Vectored-Valued Images

5.1.1 Theory

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 [5], or use LUV color space instead, or from some geometric considerations [28]. For some vision problems such as denoising and inpainting, the above mentioned methods may be effective. However, for more complex problems, such as Color2Gray [11], i.e., keeping the contrast among nearby regions when converting color images to grayscale ones, and demosaicking, 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.

Based on the theory presented in Section 3, which is for grayscale images, having PDEs for some color image processing problems becomes trivial, because we can easily extend the framework in Section 3 for learning a system of PDEs for vector-valued images. With the extended framework, the correlation among different channels of images can be automatically (but implicitly) modelled. The modifications on the framework in Section 3 include:

1. A single output channel O now becomes multiple channels: O_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:

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

   where f_r, f_s and f_m could be the indicator function ρ 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.

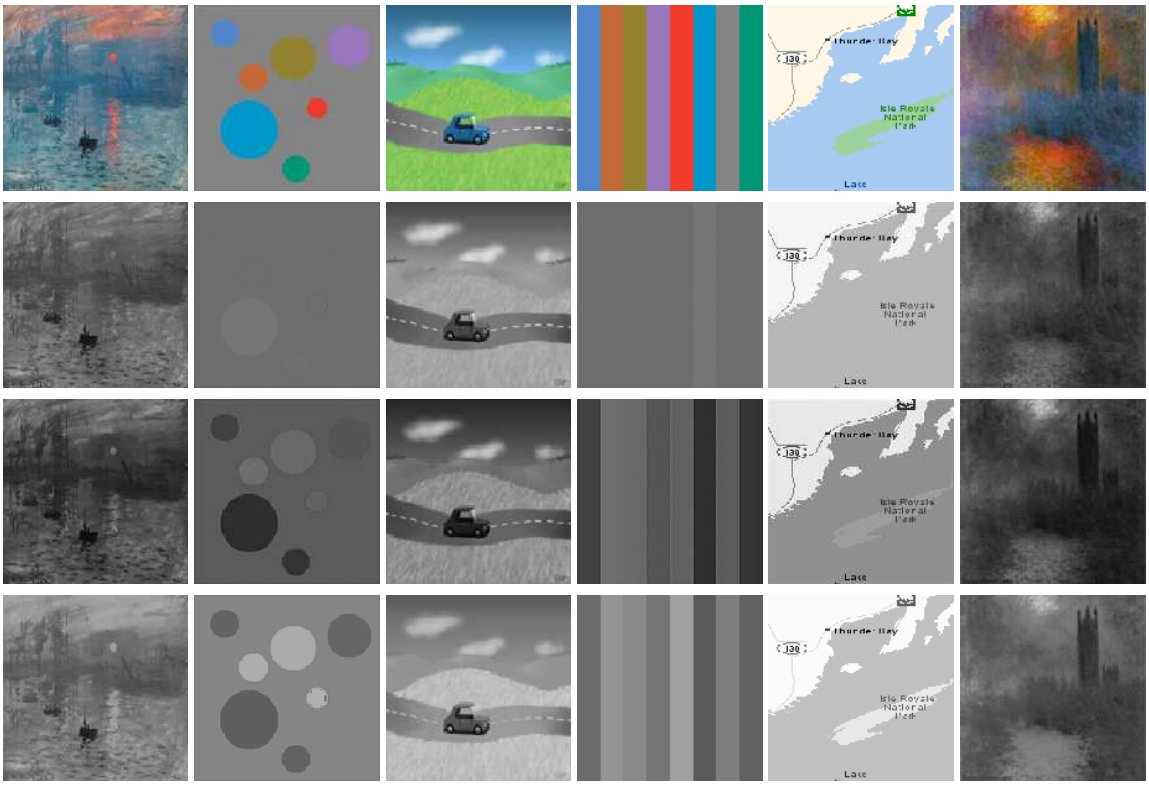


Figure 11: Comparison of the results of Photoshop Grayscale (second row), our PDEs (third row) and Color2Gray [11] (fourth row). The first row are the original color images. These images are best viewed on screen.

4. The objective functional is the sum of J's in (9) for every channel (with 16 changed to 68).

Note that for vectored-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.

5.1.2 Experiment

We test our extended framework on the Color2Gray problem [11]. Contrast among nearby regions is often lost when color images are converted to grayscale by naively computing their luminance (e.g., using using Adobe Photoshop Grayscale mode). Gooch et al. [11] 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 [11]. These 50 input/output image pairs are the training examples of our learning-based PDEs. We test the learnt PDEs on images in [11] and their websites[7]. All training and testing images are resized to 150×150. Some results are shown in Figure 11.[8] 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.

5.2 Multi-Layer PDE Systems

5.2.1 Theory

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

[7] http://www.cs.northwestern.edu/∼ago820/color2gray/

[8] Due to resizing, the results of Color2Gray in Figure 11 are slightly different from those in [11].

Full image Zoomed region BI SA [18] DFAPD [20] BI + 13 layers

Figure 12: 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 rows are the demosaicing results of different methods. These images are best viewed on screen.

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[9]. 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 [13, 3], which are composed of multiple levels of nonlinear

[9]We normalize the grayscales to $[0, 1]$ when computing.

operations, are proposed for learning highly varying functions in vision and other artificial intelligence tasks [4]. 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.

5.2.2 Experiment

We test this framework on image demosaicking. 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 full-color images. This technique is called demosaicing. The most commonly used CFAs are Bayer pattern [12, 18, 20]. 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., [12, 18, 20]). We show that with learning-based PDEs, demosaicing also becomes easy.

We used the Kodak image database[10] for the experiment. Images 1-12 are used for training

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

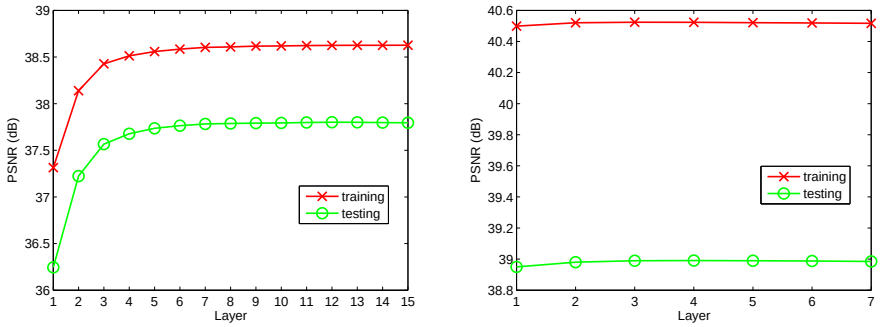
Table 3: Comparison on PSNRs of different demosaicing algorithms. "BI + 1 layer" denotes the results of single-layer PDEs using bilinearly interpolated color images as the initial functions. Other abbreviations carry the similar meaning.

	BI	AP [12]	SA [18]	DFAPD [20]
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

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 [12, 18, 20] 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 Figure 12). Learning-based PDEs can also improve the existing demosaicing algorithms by using their output as our input images (see Table 3 for an example on DFAPD [20]). Figure 13 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.

6 Conclusions and Future Work

In this paper, we have presented a framework of using PDEs as a general regressor to approximate the nonlinear mappings of different vision tasks. The experimental results on a number of vision problems show that our theory is very promising. Particularly, we obtained PDEs for some problems where PDE methods have never been tried, due to their difficulty



BI + Multi-Layer DFAPD + Multi-Layer

Figure 13: Average PSNRs of each layer's output on training and testing images.

in mathematical description. 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 vision tasks to find out to what extent it works. Second, an obvious limitation of learning-based PDEs is that all the training and testing images should be of the same size or resolution and with objects of interest roughly at the same size. It is important to consider how to learn PDEs for differently sized/scaled images. Third, the time required to learn the PDEs is very long. Although the computation can easily be parallelized, better mechanisms, e.g., better initialization, should be explored. 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. And last, it is very important to explore how to further improve the approximation power of learning-based PDEs, beyond the multi-layer formulation. We hope that someday learning-based PDEs, in their improved formulations, could be a general framework for modeling most vision problems.

References

[1] Corel photo library, corel corp., ottawa, ont., canada k1z 8r7.

[2] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing*. Springer-Verlag, 2002.

[3] Y. Bengio, P. Lamblin, P. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2006.

[4] Y. Bengio and Y. Le Cun. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. MIT Press, 2007.

[5] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH*, pages 417–424, 2000.

[6] F. Cao. *Geometric Curve Evolution and Image Processing, Lecture Notes in Mathematics, No. 1805, Springer-Verlarg*. 2003.

[7] Vicent Caselles, Jean-Michel Morel, Guillermo Sapiro, and 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.

[8] A. Friedman. *Partial Differential Equations of Parabolic Type, Prentice-Hall*. 1964.

[9] D. Gabor. Information theory in electron microscopy. *Laboratory Investigation*, 14:801–807, 1965.

[10] G. Gilboa, N. Sochen, and Y.Y. Zeevi. Image enhancement and denoising by complex diffusion processes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(8):1020–1036, 2004.

[11] Amy A. Gooch, Sven C. Olsen, Jack Tumblin, and Bruce Gooch. Color2gray: salience-preserving color removal. *ACM Transactions on Graphics*, 24(3):634–639, August 2005.

[12] B. K. Gunturk, Y. Altunbask, and R. M. Mersereau. Color plane interpolation using alternating projections. *IEEE Trans. Image Process.*, 11(9):997C1013, 2002.

[13] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[14] A.K. Jain. Partial differential equations and finite-difference methods in image processing, part 1. *Journal of Optimization Theory and Applications*, 23:65–91, 1977.

[15] B. Kimia, A. Tannenbaum, and S.W. Zucker. On optimal control methods in computer vision and image processing, 1994. in B. ter Haar Romeny (ed.) Geometry-Driven Diffusion in Computer Vision, Kluwer Academic Publishers.

[16] J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[17] C. Li, C. Xu, C. Gui, and M. Fox. Level set evolution without re-initialization: A new variational formulation. In *Proc. Computer Vision and Pattern Recognition*, 2005.

[18] X. Li. Demosaicing by successive approximations. *IEEE Trans. Image Process.*, 14(2):267–278, 2005.

[19] J.-L. Lions. *Optimal Control Systems Governed by Partial Differential Equations*. Springer-Verlarg, 1971.

[20] Daniele Menon, Stefano Andriani, and Giancarlo Calvagno. Demosaicing with directional filtering and a posteriori decision. *IEEE Trans. Image Process.*, 16(1):132–141, 2007.

[21] P. Olver. *Applications of Lie Groups to Differential Equations, Springer-Verlarg*. 1993.

[22] S.J. Osher and L. I. Rudin. Feature-oriented image enhancement using shock filters. *SIAM J. Numerical Analysis*, 27(4):919–940, 1990.

[23] Nicolas Papadakis, Thomas Corpetti, and Etienne Mémin. Dynamically consistent optical flow estimation. In *Proc. Intn'l Conf. Computer Vision*, 2007.

[24] Nicolas Papadakis and Etienne Mémin. Variational optimal control technique for the tracking of deformable objects. In *Proc. Intn'l Conf. Computer Vision*, 2007.

[25] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[26] G. Sapiro. *Geometric Partial Differential Equations and Image Analysis*. Cambridge University Press, 2001.

[27] Bart M. ter Haar Romeny. *Geometry-Driven Diffusion in Computer Vision*. Kluwer Academic Publishers, 1994.

[28] D. Tschumperlé and R. Deriche. Vector-valued image regularization with PDE's: A common framework for different applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[29] A. Witkin. Scale-space filtering. In *Proc. Int. Joint Conf. Artificial Intelligence*, 1983.

Appendix 1: Proof of (3)-(4)

Suppose Φ is the operator that maps u to the solution f to PDE (2), i.e., $f = \Phi(u)$, and given u and its perturbation δu, we define

$$r = \lim_{\varepsilon \to 0} \frac{\Phi(u + \varepsilon \cdot \delta u) - \Phi(u)}{\varepsilon}.$$

Then we have

$$\begin{aligned}
\left(\frac{\mathrm{D}J}{\mathrm{D}u}, \delta u\right)_Q &= \lim_{\varepsilon \to 0} \frac{J(\Phi(u + \varepsilon \cdot \delta u), u + \varepsilon \cdot \delta u) - J(\Phi(u), u)}{\varepsilon} \\
&= \lim_{\varepsilon \to 0} \frac{J(f + \varepsilon \cdot r + o(\varepsilon), u + \varepsilon \cdot \delta u) - J(f, u)}{\varepsilon} \\
&= \int_Q \left(g_{\langle f \rangle}(\langle u \rangle, \langle f \rangle)[r] + g_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[\delta u]\right) \mathrm{d}Q \\
&= \left(g_{\langle f \rangle}(\langle u \rangle, \langle f \rangle)[r], 1\right)_Q + \left(g_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[\delta u], 1\right)_Q \\
&= \left(g^*_{\langle f \rangle}(\langle u \rangle, \langle f \rangle)[1], r\right)_Q + \left(g^*_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[1], \delta u\right)_Q,
\end{aligned} \tag{11}$$

where $g^*_{\langle f \rangle}$ and $g^*_{\langle u \rangle}$ are the adjoint operators of $g_{\langle f \rangle}$ and $g_{\langle u \rangle}$, respectively. If we define φ as the solution to the adjoint equation (4), then we can prove that

$$\left(g^*_{\langle f \rangle}(\langle u \rangle, \langle f \rangle)[1], r\right)_Q = \left(\delta u, L^*_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[\varphi]\right)_Q. \tag{12}$$

The proof is deferred to the end of this appendix. Combining the above with (11), we have that

$$\left(\frac{\mathrm{D}J}{\mathrm{D}u}, \delta u\right)_Q = \left(g^*_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[1] + L^*_{\langle u \rangle}(\langle u \rangle, \langle f \rangle)[\varphi], \delta u\right)_Q,$$

and hence

$$\frac{\mathrm{D}J}{\mathrm{D}u} = g^*_{\langle u\rangle}(\langle u\rangle, \langle f\rangle)[1] + L^*_{\langle u\rangle}(\langle u\rangle, \langle f\rangle)[\varphi].$$

Now we prove (12). From (2) we have

$$((\Phi(u))_t, \varphi)_Q = (L(\langle u\rangle, \langle \Phi(u)\rangle), \varphi)_Q\,, \quad \forall\varphi. \tag{13}$$

and

$$((\Phi(u+\varepsilon\cdot\delta u))_t, \varphi)_Q = (L(\langle u+\varepsilon\cdot\delta u\rangle, \langle \Phi(u+\varepsilon\cdot\delta u)\rangle), \varphi)_Q\,, \quad \forall\varphi. \tag{14}$$

Subtract (14) with (13), we have that (note that $f = \Phi(u)$):

$$\begin{aligned} & ((\Phi(u+\varepsilon\cdot\delta u) - \Phi(u))_t, \varphi)_Q \\ = \;& (L(\langle u+\varepsilon\cdot\delta u\rangle, \langle \Phi(u+\varepsilon\cdot\delta u)\rangle) - L(\langle u\rangle, \langle \Phi(u)\rangle), \varphi)_Q \\ = \;& \big(\varepsilon\cdot L_{\langle u\rangle}(\langle u\rangle, \langle f\rangle[\delta u] + L_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[\Phi(u+\varepsilon\cdot\delta u) - \Phi(u)], \varphi\big) + o(\varepsilon), \quad \forall\varphi. \end{aligned}$$

Dividing both sides with ε and let $\varepsilon \to 0$, we see that

$$(r_t, \varphi)_Q = \big(L_{\langle u\rangle}(\langle u\rangle, \langle f\rangle)[\delta u], \varphi\big)_Q + \big(L_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[r], \varphi\big)_Q\,, \quad \forall\varphi. \tag{15}$$

Integrate by parts, we have:

$$(r, \varphi)_\Omega|_0^T - (r, \varphi_t)_Q = \big(\delta u, L^*_{\langle u\rangle}(\langle u\rangle, \langle f\rangle)[\varphi]\big)_Q + \big(r, L^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[\varphi]\big)_Q\,, \quad \forall\varphi. \tag{16}$$

Now that φ satisfies (4), we have that

$$(r, \varphi)_\Omega|_0^T = 0, \text{ and } \big(g^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[1], r\big)_Q = \big(-\varphi_t - L^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[\varphi], r\big)_Q\,.$$

Combining the above with (15), we arrive at:

$$\big(g^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[1], r\big)_Q = \big(-\varphi_t - L^*_{\langle f\rangle}(\langle u\rangle, \langle f\rangle)[\varphi], r\big)_Q = \big(\delta u, L^*_{\langle u\rangle}(\langle u\rangle, \langle f\rangle)[\varphi]\big)_Q\,.$$

Appendix 2: 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 (7), i.e.,

$$\begin{aligned}\frac{\partial O(\mathbf{x}-\mathbf{x}_0)}{\partial t} &= \tilde{L}_O(\langle O(\mathbf{x}-\mathbf{x}_0)\rangle, \langle\rho(\mathbf{x}-\mathbf{x}_0)\rangle, \mathbf{x}, t)\\ &= \tilde{L}_O(\langle O\rangle(\mathbf{x}-\mathbf{x}_0), \langle\rho\rangle(\mathbf{x}-\mathbf{x}_0), \mathbf{x}, t).\end{aligned} \tag{17}$$

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

$$\frac{\partial O(\mathbf{x})}{\partial t} = \tilde{L}_O(\langle O\rangle(\mathbf{x}), \langle\rho\rangle(\mathbf{x}), \mathbf{x}+\mathbf{x}_0, t). \tag{18}$$

On the other hand, the pair $(\rho(\mathbf{x}), O(\mathbf{x}))$ also fulfils (7), i.e.,

$$\frac{\partial O(\mathbf{x})}{\partial t} = \tilde{L}_O(\langle O\rangle(\mathbf{x}), \langle\rho\rangle(\mathbf{x}), \mathbf{x}, t). \tag{19}$$

Therefore,

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

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

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

□

Appendix 3: The Gâteaux Derivatives

We use (6) to compute the Gâteaux derivatives. The Lagrangian function is:

$$\begin{aligned}&\tilde{J}(\{O_m\}_{m=1}^M, \{a_j\}_{j=0}^{16}, \{b_j\}_{j=0}^{16}; \{\varphi_m\}_{m=1}^M, \{\phi_m\}_{m=1}^M)\\ =\ & J(\{O_m\}_{m=1}^M, \{a_j\}_{j=0}^{16}, \{b_j\}_{j=0}^{16}) + \sum_{m=1}^M \int_Q \varphi_m \left[(O_m)_t - L_O(\mathbf{a}, \langle O_m\rangle, \langle\rho_m\rangle)\right] \mathrm{d}Q\\ &+ \sum_{m=1}^M \int_Q \phi_m \left[(\rho_m)_t - L_\rho(\mathbf{b}, \langle\rho_m\rangle, \langle O_m\rangle)\right] \mathrm{d}Q,\end{aligned} \tag{20}$$

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:

$$\begin{aligned}
& L_O(\mathbf{a}, \langle O + \varepsilon \cdot \delta O\rangle, \langle\rho\rangle) - L_O(\mathbf{a}, \langle O\rangle, \langle\rho\rangle) \\
= \; & \varepsilon \cdot \left(\frac{\partial L_O}{\partial O}(\delta O) + \frac{\partial L_O}{\partial O_x}\frac{\partial(\delta O)}{\partial x} + \cdots + \frac{\partial L_O}{\partial O_{yy}}\frac{\partial^2(\delta O)}{\partial y^2} \right) + o(\varepsilon) \\
= \; & \varepsilon \sum_{p\in\wp} \frac{\partial L_O}{\partial O_p}\frac{\partial^{|p|}(\delta O)}{\partial p} + o(\varepsilon) \\
= \; & \varepsilon \sum_{p\in\wp} \sigma_{O;p}\frac{\partial^{|p|}(\delta O)}{\partial p} + o(\varepsilon), \\
& L_\rho(\mathbf{b}, \langle\rho\rangle, \langle O + \varepsilon \cdot \delta O\rangle) - L_\rho(\mathbf{b}, \langle\rho\rangle, \langle O\rangle) \\
= \; & \varepsilon \sum_{p\in\wp} \sigma_{\rho;p}\frac{\partial^{|p|}(\delta O)}{\partial p} + o(\varepsilon),
\end{aligned} \tag{21}$$

where

$$\sigma_{O;p} = \frac{\partial L_O}{\partial O_p} = \sum_{i=0}^{16} a_i \frac{\partial \mathrm{inv}_i(\rho, O)}{\partial O_p}, \quad \text{and} \quad \sigma_{\rho;p} = \frac{\partial L_\rho}{\partial O_p} = \sum_{i=0}^{16} b_i \frac{\partial \mathrm{inv}_i(O, \rho)}{\partial O_p}.$$

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

$$\begin{aligned}
\delta\tilde{J}_k = \; & \tilde{J}(\cdots, O_k + \varepsilon \cdot \delta O_k, \cdots) - \tilde{J}(\cdots, O_k, \cdots) \\
= \; & \frac{1}{2}\int_\Omega \left[\Big((O_k + \varepsilon \cdot \delta O_k)(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \Big)^2 - \Big(O_k(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \Big)^2 \right] \mathrm{d}\Omega \\
& + \int_Q \varphi_k \left\{ [(O_k + \varepsilon \cdot \delta O_k)_t - L_O(\mathbf{a}, \langle O_k + \varepsilon \cdot \delta O_k\rangle, \langle\rho_k\rangle) \right. \\
& \left. - [(O_k)_t - L_O(\mathbf{a}, \langle O_k\rangle, \langle\rho_k\rangle)] \right\} \mathrm{d}Q \\
& + \int_Q \phi_k \left\{ [(\rho_k)_t - L_\rho(\mathbf{b}, \langle\rho_k\rangle, \langle O_k + \varepsilon \cdot \delta O_k\rangle)] \right. \\
& \left. - [(\rho_k)_t - L_\rho(\mathbf{b}, \langle\rho_k\rangle, \langle O_k\rangle)] \right\} \mathrm{d}Q \\
= \; & \varepsilon \int_\Omega \Big(O_k(\mathbf{x}, 1) - \tilde{O}_k(\mathbf{x}) \Big) \delta O_k(\mathbf{x}, 1) \mathrm{d}\Omega + \varepsilon \int_Q \varphi_k (\delta O_k)_t \mathrm{d}Q \\
& - \varepsilon \int_Q \varphi_k \sum_{p\in\wp} \sigma_{O;p} \frac{\partial^{|p|}(\delta O_k)}{\partial p} \mathrm{d}Q - \varepsilon \int_Q \phi_k \sum_{p\in\wp} \sigma_{\rho;p} \frac{\partial^{|p|}(\delta O_k)}{\partial p} \mathrm{d}Q + o(\varepsilon).
\end{aligned} \tag{22}$$

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

$$\begin{aligned}
\delta\tilde{J}_k = & \ \varepsilon\int_\Omega \Big(O_k(\mathbf{x},1)-\tilde{O}_k(\mathbf{x})\Big)\,\delta O_k(\mathbf{x},1)\mathrm{d}\Omega+\varepsilon\int_\Omega(\varphi_k\cdot\delta O_k)(\mathbf{x},1)\mathrm{d}\Omega \\
& -\varepsilon\int_Q(\varphi_k)_t\delta O_k\mathrm{d}Q-\varepsilon\int_Q\sum_{p\in\wp}(-1)^{|p|}\frac{\partial(\sigma_{O;p}\varphi_k)}{\partial p}\delta O_k\mathrm{d}Q \\
& -\varepsilon\int_Q\sum_{p\in\wp}(-1)^{|p|}\frac{\partial^{|p|}(\sigma_{\rho;p}\phi_k)}{\partial p}\delta O_k\mathrm{d}Q+o(\varepsilon) \\
= & \ \varepsilon\int_Q\Big[\Big(\varphi_k+O_k(\mathbf{x},1)-\tilde{O}_k(\mathbf{x})\Big)\,\delta(t-1) \\
& \quad -(\varphi_k)_t-(-1)^{|p|}\frac{\partial^{|p|}(\sigma_{O;p}\varphi_k+\sigma_{\rho;p}\phi_k)}{\partial p}\Big]\,\delta O_k\mathrm{d}Q+o(\varepsilon).
\end{aligned} \tag{23}$$

So the adjoint equation for φ_k is

$$\begin{cases}
\dfrac{\partial\varphi_m}{\partial t}+\displaystyle\sum_{p\in\wp}(-1)^{|p|}\left(\sigma_{O;p}\varphi_m+\sigma_{\rho;p}\phi_m\right)_p=0, & (\mathbf{x},t)\in Q, \\
\varphi_m=0, & (\mathbf{x},t)\in\Gamma, \\
\varphi_m|_{t=1}=\tilde{O}_m-O_m(1), & \mathbf{x}\in\Omega,
\end{cases} \tag{24}$$

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

$$\begin{cases}
\dfrac{\partial\phi_m}{\partial t}+\displaystyle\sum_{p\in\wp}(-1)^{|p|}\left(\tilde{\sigma}_{O;p}\varphi_m+\tilde{\sigma}_{\rho;p}\phi_m\right)_p=0, & (\mathbf{x},t)\in Q, \\
\phi_m=0, & (\mathbf{x},t)\in\Gamma, \\
\phi_m|_{t=1}=0, & \mathbf{x}\in\Omega,
\end{cases} \tag{25}$$

where

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

Also by perturbation, it is easy to check that:

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

So by (6), the above are also the Gâteaux derivatives of J with respect to a_i and b_i, respectively.