LOW DIMENSIONAL MANIFOLD MODEL FOR IMAGE PROCESSING *

STANLEY OSHER †, ZUOQIANG SHI ‡, AND WEI ZHU §

Abstract. In this paper, we propose a novel low dimensional manifold model (LDMM) and apply it to some image processing problems. LDMM is based on the fact that the patch manifolds of many natural images have low dimensional structure. Based on this fact, the dimension of the patch manifold is used as a regularization to recover the image. The key step in LDMM is to solve a Laplace-Beltrami equation over a point cloud which is solved by the point integral method. The point integral method enforces the sample point constraints correctly and gives better results than the standard graph Laplacian. Numerical simulations in image denoising, inpainting and super-resolution problems show that LDMM is a powerful method in image processing.

1. Introduction. Many image processing problems can be formalized as the recovery of an image $f \in \mathbb{R}^{m\times n}$ from a set of noisy linear measurements

$$y = \Phi f + \varepsilon \tag{1.1}$$

where ε is the noise. The operator, Φ, typically accounts for some damage to the image, for instance, blurring, missing pixels or downsampling, so that the measured data y only captures a small portion of the original image f. It is an ill-posed problem to recover the original image from partial information. In order to solve this ill-posed problem, one needs to have some prior knowledge of the image. Usually, this prior information gives different regularizations. With the help of regularizations, many image processing problems are formulated as optimization problems.

One of the most widely used regularizations is total variation, introduced by Rudin, Osher and Fatemi (ROF)[12]. In the ROF model, the following optimization problem is solved.

$$\min_f \|f\|_{TV} + \frac{\mu}{2}\|y - \Phi f\|_{L^2}^2 \tag{1.2}$$

where $\|f\|_{TV} = \int |\nabla f(x)| \mathrm{d}x$. With the Bregman techniques [9, 6], TV regularized problems can be solved efficiently. It is well known that TV based model can restore the "cartoon" part of the image very well, while the performance on the texture part of the image is not so good.

The nonlocal methods are another class widely used in image processing. These were first proposed by Buades, Coll, and Morel [1, 2] as a nonlocal filter for image denoising and were later formulated in a variational framework by Gilboa and Osher [4, 5]. In the nonlocal method, the local derivatives are replaced by their nonlocal counterpart:

$$\nabla_w u(x, y) = \sqrt{w(x, y)}(u(x) - u(y)) \tag{1.3}$$

*Research supported by DOE-SC0013838 and NSF DMS-1118971. Z. Shi was partially supported by NSFC Grant 11371220.

†Department of Mathematics, University of California, Los Angeles, CA 90095, USA, *Email: sjo@math.ucla.edu*

‡Yau Mathematical Sciences Center, Tsinghua University, Beijing, China, 100084. *Email: zqshi@math.tsinghua.edu.cn.*

§Department of Mathematics, University of California, Los Angeles, CA 90095, USA, *Email: weizhu731@math.ucla.edu*

where w is a weight function defined as

$$w(x,y) = \exp\left(-\int G(s)|u(x+s) - u(y+s)|^2 ds\right), \tag{1.4}$$

G is a Gaussian. Using the nonlocal derivatives, the nonlocal total variation model is given as following

$$\min_f \|\nabla_w f\|_{L^1} + \frac{\mu}{2}\|y - \Phi f\|_{L^2}^2, \tag{1.5}$$

where

$$\|\nabla_w(f)\|_{L^1} = \sum_x \left(\sum_y w(x,y)(f(x) - f(y))^2\right)^{1/2}. \tag{1.6}$$

It has been shown that the nonlocal model recovers textures well.

In this paper, inspired by the nonlocal method and the manifold model of image [11], we proposed a low dimensional manifold model (LDMM) for image processing. Consider a $m \times n$ size image $f \in \mathbb{R}^{m\times n}$. For any pixel (i,j), where $1 \le i \le m, 1 \le j \le n$, let this pixel constitute the left-top pixel in a $s_1 \times s_2$ patch and denote this patch as p_{ij}. Let $\mathcal{P}(f)$ denote the collection of all such patches, i.e.,

$$\mathcal{P}(f) = \{p_{i,j} : (i,j) \in \Theta \subset \{1,2,\cdots,m\} \times \{1,2,\cdots,n\}\}. \tag{1.7}$$

where Θ is a index set such that the union of the patch set $\mathcal{P}(f)$ covers the whole image. There are many ways to choose Θ. For example, we can choose $\Theta = \{1,2,\cdots,m\} \times \{1,2,\cdots,n\}$ or $\Theta = \{1, s_1+1, 2s_1+1, \cdots, m\} \times \{1, s_2+1, 2s_2+1, \cdots, n\}$. This freedom may be used to accelerate the computation in LDMM.

In this paper, $\mathcal{P}(f)$ is called the patch set of f. $\mathcal{P}(f)$ can be seen as a point set in $\mathbb{R}^d$ with $d = s_1 s_2$. The basic assumption in this paper is that $\mathcal{P}(f)$ samples a low-dimensional smooth manifold $\mathcal{M}(f)$ embedded in $\mathbb{R}^d$, which is called the patch manifold of f. It was revealed that for many classes of images, this assumption holds [10, 11] Based on this assumption, one natural regularization involves the dimension of the patch manifold. We want to recover the original image such that the dimension of its patch manifold is as small as possible. This idea formally gives the following optimization problem:

$$\min_f \dim(\mathcal{M}(f)) + \lambda\|y - \Phi f\|_2^2. \tag{1.8}$$

The problem remaining is how to compute $\dim(\mathcal{M}(f))$ for given image f. Fortunately, using some basic tools in differential geometry, we find that the dimension of a smooth manifold embedded in $\mathbb{R}^d$ can be calculated by a simple formula

$$\dim(\mathcal{M}) = \sum_{j=1}^{d} |\nabla_{\mathcal{M}} \alpha_i(\boldsymbol{x})|^2$$

where α_i is the coordinate function, for any $\boldsymbol{x} = (x_1, \cdots, x_d) \in \mathcal{M} \subset \mathbb{R}^d$,

$$\alpha_i(\boldsymbol{x}) = x_i. \tag{1.9}$$

Using this formula, the optimization problem (1.8) can be reformulated as

$$\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})} + \lambda\|y-\Phi f\|^2_{l^2}, \qquad \text{subject to:} \quad \mathcal{P}(f)\subset\mathcal{M}. \tag{1.10}$$

In this paper, the optimization problem (1.10) is solved by an alternating direction iteration. First, we fix the manifold $\mathcal{M}$, and update the image f. Then the image is fixed and we update the manifold. This process is repeated until convergence. In this two step iteration, the second step is relatively easy. It is done by directly applying the patch operator on the image f.

The first step is more difficult. To update the image, we need to solve Laplace-Beltrami equations over the manifold.

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{x}) + \mu \sum_{\boldsymbol{y}\in\Omega} \delta(\boldsymbol{x}-\boldsymbol{y})(u(\boldsymbol{y})-v(\boldsymbol{y})) = 0, & \boldsymbol{x}\in\mathcal{M} \\ \dfrac{\partial u}{\partial \mathbf{n}}(\boldsymbol{x}) = 0, & \boldsymbol{x}\in\partial\mathcal{M}. \end{cases} \tag{1.11}$$

where $\mathcal{M}$ is a manifold, $\partial\mathcal{M}$ is the boundary of $\mathcal{M}$, $\mathbf{n}$ is the out normal of $\partial\mathcal{M}$. δ is the Dirac-δ function in $\mathcal{M}$, v is a given function in $\mathcal{M}$.

The explicit form of the manifold $\mathcal{M}$ is not known. We only know a set of unstructured points which samples the manifold $\mathcal{M}$ in high dimensional Euclidean space. It is not easy to solve this Laplace-Beltrami equation over this unstructured high dimensional point set. A graph Laplacian is usually used to approximate the Laplace-Beltrami operator on point cloud. However, from the point of view of numerical PDE, the graph Laplacian was found to be an inconsistent method due to the lack of boundary correction. This is also confirmed by our numerical simulations, e.g. (Fig. 1(d)).

Instead, in this paper, we use the point integral method (PIM) [8, 14, 15] to solve the Laplace-Beltrami equation over the point cloud. In PIM, the discretized linear system of the Laplace-Beltrami equation (1.11) is given as following:

$$\frac{|\mathcal{M}|}{N}\sum_{j=1}^{N} R_t(\boldsymbol{x}_i,\boldsymbol{x}_j)(u_i-u_j) + \mu t\sum_{j=1}^{N}\bar{R}_t(\boldsymbol{x}_i,\boldsymbol{x}_j)(u_j-v_j) = 0 \tag{1.12}$$

where $\{\boldsymbol{x}_1,\cdots,\boldsymbol{x}_N\}$ samples $\mathcal{M}$, $v_j = v(\boldsymbol{x}_j)$ and $|\mathcal{M}|$ is the volume of the manifold $\mathcal{M}$. R_t and $\bar{R}_t$ are kernel functions which are given in Section 4.1.

Compared with the graph Laplacian which is widely used in the machine learning and nonlocal methods, a boundary term is added in PIM based on an integral approximation of the Laplace-Beltrami operator. With the help of this boundary term, the values in the retained pixels correctly spread to the missing pixels which gives much better recovery (Fig. 1(c)).

The rest of the paper is organized as following. The patch manifold is analyzed in Section 2. Several examples are given to show that it usually has a low dimension structure. In Section 3, we introduce the low dimensional manifold model. The numerical methods including the point integral method are discussed in Section 4. This is the most important section for any potential user of our method. In Section 5, we compare the performance of the LDMM method and classical nonlocal methods and carefully explain the apparently slight differences between the two methods. Numerical results are shown in Section 6. Concluding remarks are made in Section 7.

2. Patch manifold. In this section, we analyze the patch manifold and give several examples. We consider a discrete image $f \in \mathbb{R}^{m\times n}$ and assume that there is a continuous counterpart $\tilde{f} \in L^2([0,1]^2)$ such that $f = I(\tilde{f})$, where I is a "camera" operator which maps a 2D function to a discrete image. For any $x \in [0,1]^2$, a patch $p_x(\tilde{f})$ of size (τ_1, τ_2) is a function on $[0,\tau_1] \times [0,\tau_2]$,

$$p_x(\tilde{f})(s) = \tilde{f}(x+s), \quad \forall s \in [0,\tau_1] \times [0,\tau_2]. \tag{2.1}$$

The patch manifold is defined as

$$\mathcal{M}(f) = \{I(p_x(\tilde{f})) : x \in [0,1]^2\} \subset \mathbb{R}^d, \quad d = s_1 s_2. \tag{2.2}$$

It is obvious that the patch set $\mathcal{P}(f) \subset \mathcal{M}(f)$ is a sample of $\mathcal{M}$, where the patch set $\mathcal{P}(f)$ is defined in (1.7).

The patch set and patch manifold have been well studied in the literature. Lee et al. studied the patch set for discretized images with 3×3 patches [7]. Their results were refined later by Carlsson et al. [3] that perform a simplicial approximation of the manifold. Peyré studied several models of local image manifolds for which an explicit parameterization is available [10, 11]. One of the most important feature of the patch manifold is that it is close to a low-dimensional manifold for many natural images. Now, let us see several simple examples.

If f is a C^2 function which correspond to a smooth image. Using Taylor's expansion, $p_x(f)$ can be well approximated by a linear function

$$p_x(f)(y) \approx f(x) + (y-x) \cdot \nabla f(x). \tag{2.3}$$

This fact implies that $\mathcal{M}(f)$ is close to a 3D manifold.

If f is a piecewise constant function which corresponds to a cartoon image, then, any patch, $p_x(f)$, is well approximated by a straight edge patch. Each patch is parametrized by the location and the orientation of the edge. This suggests that for a piecewise constant function, $\mathcal{M}(f)$ is also close to a 3D manifold.

If f is a oscillatory function corresponding to a texture image. We assume that f can be represented as

$$f(x) \approx a(x) \cos \theta(x). \tag{2.4}$$

where $a(x)$ and $\theta(x)$ are both smooth functions. For each pixel x, the patch $p_x(f)$ can be well approximated by

$$p_x(f) \approx a_L(y) \cos \theta_L(y), \tag{2.5}$$

where $a_L(x)$ and $\theta_L(x)$ are linear approximations of $a(x)$ and $\theta(x)$ at x, i.e.,

$$a_L(y) = a(x) + (y-x) \cdot \nabla a(x), \quad \theta_L(y) = \theta(x) + (y-x) \cdot \nabla \theta(x).$$

This means that the patch manifold $\mathcal{M}(f)$ is approximately a 6-dimensional manifold.

These simple examples show that for many images, smooth, cartoon and texture, the patch manifold is approximately a low dimensional manifold. Then one natural idea is to retrieve the original image by looking for the patch manifold with the lowest dimension. This idea leads to a low dimensional manifold model introduced in the next section.

3. Low dimensional manifold model. Based on the discussion in the previous section, we know that an important feature of the patch manifold is low dimensionality. One natural idea is to use the dimension of the patch manifold as the regularization to recover the original image. In the low dimensional manifold model, we want to recover the image f such that the dimension of its patch manifold $\mathcal{M}(f)$ is as small as possible. This idea formally gives an optimization problem:

$$\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^d}} \dim(\mathcal{M}), \qquad \text{subject to:} \quad y=\Phi f+\varepsilon, \quad \mathcal{P}(f)\subset\mathcal{M} \tag{3.1}$$

where $\dim(\mathcal{M})$ is the dimension of the manifold $\mathcal{M}$.

However, this optimization problem is not mathematically well defined, since we do not know how to compute $\dim(\mathcal{M})$ with given $\mathcal{P}(f)$. Next, we will derive a simple formula for $\dim(\mathcal{M})$ using some basic knowledge of differential geometry.

3.1. Calculation of $\dim(\mathcal{M})$. Here we assume $\mathcal{M}$ is a smooth manifold embedded in $\mathbb{R}^d$. First, we introduce some notation. Since $\mathcal{M}$ is a smooth submanifold isometrically embedded in $\mathbb{R}^d$, it can be locally parametrized as follows,

$$\boldsymbol{x}=\psi(\boldsymbol{\gamma}) : U\subset\mathbb{R}^k\to\mathcal{M}\subset\mathbb{R}^d \tag{3.2}$$

where $k=\dim(\mathcal{M})$, $\boldsymbol{\gamma}=(\gamma^1,\cdots,\gamma^k)^t\in\mathbb{R}^k$ and $\boldsymbol{x}=(x^1,\cdots,x^d)^t\in\mathcal{M}$.

Let $\partial_{i'}=\frac{\partial}{\partial\gamma^{i'}}$ be the tangent vector along the direction $\gamma^{i'}$. Since $\mathcal{M}$ is a submanifold in $\mathbb{R}^d$ with induced metric, $\partial_{i'}=(\partial_{i'}\psi^1,\cdots,\partial_{i'}\psi^d)$ and the metric tensor

$$g_{i'j'}=<\partial_{i'},\partial_{j'}>=\sum_{l=1}^{d}\partial_{i'}\psi^l\partial_{j'}\psi^l. \tag{3.3}$$

Let $g^{i'j'}$ denote the inverse of $g_{i'j'}$, i.e.,

$$\sum_{l'=1}^{k} g_{i'l'}\,g^{l'j'}=\delta_{i'j'}=\begin{cases}1, & i'=j',\\ 0, & i'\neq j'.\end{cases} \tag{3.4}$$

For any function u on $\mathcal{M}$, let $\nabla_{\mathcal{M}}u$ denote the gradient of u on $\mathcal{M}$,

$$\nabla_{\mathcal{M}}u=\sum_{i',j'=1}^{k} g^{i'j'}\partial_{j'}u\,\partial_{i'}. \tag{3.5}$$

We can also view the gradient $\nabla_{\mathcal{M}}u$ as a vector in the ambient space $\mathbb{R}^d$ and let $\nabla^j_{\mathcal{M}}u$ denote the u component of the gradient $\nabla_{\mathcal{M}}u$ in the ambient coordinates, i.e.,

$$\nabla^j_{\mathcal{M}}u=\sum_{i',j'=1}^{k}\partial_{i'}\psi^j g^{i'j'}\partial_{j'}u, \quad j=1,\cdots,d. \tag{3.6}$$

Let $\alpha_i,\ i=1,\cdots,d$ be the coordinate functions on $\mathcal{M}$, i.e.

$$\alpha_i(\boldsymbol{x})=x_i, \quad \forall\boldsymbol{x}=(x_1,\cdots,x_d)\in\mathcal{M} \tag{3.7}$$

Then, we have the following formula

PROPOSITION 3.1. *Let $\mathcal{M}$ be a smooth submanifold isometrically embedded in $\mathbb{R}^d$. For any $\boldsymbol{x} \in \mathcal{M}$,*

$$\dim(\mathcal{M}) = \sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j(\boldsymbol{x})\|^2$$

Proof. First, following the definition of $\nabla_{\mathcal{M}}$, we have

$$\begin{aligned}
\sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2 &= \sum_{i,j=1}^{d} \nabla^i_{\mathcal{M}} \alpha_j \nabla^i_{\mathcal{M}} \alpha_j \\
&= \sum_{i,j=1}^{d} \left(\sum_{i',j'=1}^{k} \partial_{i'} \psi^j g^{i'j'} \partial_{j'} \alpha_j \right) \left(\sum_{i'',j''=1}^{k} \partial_{i''} \psi^j g^{i''j''} \partial_{j''} \alpha_j \right) \\
&= \sum_{j=1}^{d} \sum_{i',j',i'',j''=1}^{k} \left(\sum_{i=1}^{d} \partial_{i'} \psi^i \partial_{i''} \psi^i \right) g^{i'j'} g^{i''j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j \\
&= \sum_{j=1}^{d} \sum_{j',i'',j''=1}^{k} \left(\sum_{i'=1}^{k} g_{i'i''} g^{i'j'} \right) g^{i''j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j \\
&= \sum_{j=1}^{d} \sum_{j',i'',j''=1}^{k} \delta_{i''j'} g^{i''j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j \\
&= \sum_{j=1}^{d} \sum_{j',j''=1}^{k} g^{j'j''} \partial_{j'} \alpha_j \partial_{j''} \alpha_j.
\end{aligned}$$

The second equality comes from the definition of gradient on $\mathcal{M}$, (3.6). The third and fourth equalities are due to (3.3) and (3.4) respectively.

Notice that

$$\partial_{j'} \alpha_j = \frac{\partial}{\partial \gamma^{j'}} \alpha_j(\psi(\boldsymbol{\gamma})) = \partial_{j'} \psi^j. \tag{3.8}$$

It follows that

$$\begin{aligned}
\sum_{j=1}^{d} \|\nabla_{\mathcal{M}} \alpha_j\|^2 &= \sum_{j=1}^{d} \sum_{j',j''=1}^{k} g^{j'j''} \partial_{j'} \psi^j \partial_{j''} \psi^j \\
&= \sum_{j',j''=1}^{k} g^{j'j''} \left(\sum_{j=1}^{d} \partial_{j'} \psi^j \partial_{j''} \psi^j \right) \\
&= \sum_{j',j''=1}^{k} g^{j'j''} g_{j'j''} \\
&= \sum_{j'=1}^{k} \delta_{j'j'} = k = \dim(\mathcal{M})
\end{aligned} \tag{3.9}$$

□

Using above proposition, the optimization problem (3.1) can be rewritten as

$$\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \mathcal{M}\subset\mathbb{R}^d}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}}\alpha_i\|^2_{L^2(\mathcal{M})} + \lambda\|y-\Phi f\|^2_{l^2}, \qquad \text{subject to:} \quad \mathcal{P}(f)\subset\mathcal{M}. \tag{3.10}$$

where

$$\|\nabla_{\mathcal{M}}\alpha_i\|_{L^2(\mathcal{M})} = \left(\int_{\mathcal{M}} \|\nabla_{\mathcal{M}}\alpha_i(\boldsymbol{x})\|^2 \mathrm{d}\boldsymbol{x}\right)^{1/2}. \tag{3.11}$$

This is the optimization problem we need to solve.

4. Numerical method. The optimization problem (3.10) is highly nonlinear and nonconvex. In this paper, we propose an iterative method to solve it approximately. In the iteration, first the manifold is fixed, the image and the coordinate functions are computed. Then the manifold is updated using the new image and coordinate functions. More specifically, the algorithm is as following:

- With a guess of the manifold $\mathcal{M}^n$ and a guess of the image f^n satisfying $\mathcal{P}(f^n)\subset\mathcal{M}^n$, compute the coordinate functions $\alpha_i^{n+1}, i=1,\cdots,d$ and f^{n+1},

$$(f^{n+1},\alpha_1^{n+1},\cdots,\alpha_d^{n+1}) = \arg\min_{\substack{f\in\mathbb{R}^{m\times n},\\ \alpha_1,\cdots,\alpha_d\in H^1(\mathcal{M}^n)}} \quad \sum_{i=1}^{d} \|\nabla_{\mathcal{M}^n}\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \lambda\|y-\Phi f\|^2_{l^2}, \tag{4.1}$$
$$\text{subject to:} \quad \alpha_i(p_x(f^n)) = p_x^i(f),$$

  where $p_x^i(f)$ is the ith element of patch $p_x(f)$.
- Update $\mathcal{M}$ by setting

$$\mathcal{M}^{n+1} = \left\{(\alpha_1^{n+1}(\boldsymbol{x}),\cdots,\alpha_d^{n+1}(\boldsymbol{x})) : \boldsymbol{x}\in\mathcal{M}^n\right\}. \tag{4.2}$$

- repeat these two steps until convergence.

In the above iteration, the manifold is easy to update. The key step is to solve equality (4.1). Now, (4.1) is a linear optimization problem with constraints. We use Bregman iteration [9] to enforce the constraints in (4.1) which gives the following algorithm:

- Update $(f^{n+1,k+1},\boldsymbol{\alpha}^{n+1,k+1})$ by solving

$$\begin{aligned}&(f^{n+1,k+1},\alpha_1^{n+1,k+1},\cdots,\alpha_d^{n+1,k+1})\\ &= \arg\min_{\substack{\alpha_1,\cdots,\alpha_d\in H^1(\mathcal{M}^n),\\ f\in\mathbb{R}^{m\times n}}} \sum_{i=1}^{d} \|\nabla\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu\|\boldsymbol{\alpha}(\mathcal{P}(f^n)) - \mathcal{P}(f) + d^k\|^2_{\mathrm{F}} + \lambda\|y-\Phi f\|^2_2,\end{aligned}$$

  where

$$\boldsymbol{\alpha}(\mathcal{P}(f^n)) = \begin{pmatrix} \alpha_1(\mathcal{P}(f^n)) \\ \alpha_2(\mathcal{P}(f^n)) \\ \vdots \\ \alpha_d(\mathcal{P}(f^n)) \end{pmatrix} \in \mathbb{R}^{d\times N}, \quad N = |\mathcal{P}(f^n)|$$

  and $\alpha_i(\mathcal{P}(f^n)) = (\alpha_i(x))_{x\in\mathcal{P}(f^n)}, i=1\cdots,d$ are N dimensional row vectors. $\mathcal{P}(f)$ is also a $d\times N$ matrix, with each column represents one patch in $\mathcal{P}(f)$. $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm.

- Update d^{k+1},

$$d^{k+1} = d^k + \boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k+1})$$

.

To further simplify the algorithm, we use the idea of split Bregman iteration [6] to update f and α_i sequentially.

- Solve $\alpha_i^{n+1,k+1},\ i = 1, \cdots, d$ with fixed $f^{n+1,k}$,

(4.3)

$$\begin{aligned}&(\alpha_1^{n+1,k+1}, \cdots, \alpha_d^{n+1,k+1})\\ &= \arg\min_{\alpha_1,\cdots,\alpha_d \in H^1(\mathcal{M}^n)} \sum_{i=1}^{d} \|\nabla\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu\|\boldsymbol{\alpha}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k}) + d^k\|^2_{\mathrm{F}}.\end{aligned}$$

- Update $f^{n+1,k+1}$ as following

(4.4)
$$f^{n+1,k+1} = \arg\min_{f\in\mathbb{R}^{m\times n}} \quad \lambda\|y - \Phi f\|^2_{l^2} + \mu\|\boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f) + d^k\|^2_{\mathrm{F}}$$

- Update d^{k+1},

$$d^{k+1} = d^k + \boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k+1})$$

Notice that in (4.3), $\alpha_i^{n+1,k+1},\ i = 1, \cdots, d$ can be solved separately,

(4.5)
$$\alpha_i^{n+1,k+1} = \arg\min_{\alpha_i \in H^1(\mathcal{M}^n)} \quad \|\nabla\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu\|\alpha_i(\mathcal{P}(f^n)) - \mathcal{P}_i(f^{n+1,k}) + d_i^k\|^2$$

where $\mathcal{P}_i(f^n)$ is the ith row of matrix $\mathcal{P}(f^n)$.

Summarizing above discussion, we get Algorithm 1 to solve the optimization problem (3.10) in LDMM.

In Algorithm 1, the most difficult part is to solve following type of optimization problem

$$\min_{u\in H^1(\mathcal{M})} \quad \|\nabla_{\mathcal{M}} u\|^2_{L^2(\mathcal{M})} + \mu\sum_{\boldsymbol{y}\in\Omega} |u(\boldsymbol{y}) - v(\boldsymbol{y})|^2 \tag{4.9}$$

where u can be any α_i, $\mathcal{M} = \mathcal{M}^n$, $\Omega = \mathcal{P}(f^n)$ and $v(\boldsymbol{y})$ is a given function on Ω.

By a standard variational approach, we know that the solution of (4.9) can be obtained by solving the following PDE

$$\begin{cases} -\Delta_{\mathcal{M}} u(\boldsymbol{x}) + \mu\sum_{\boldsymbol{y}\in\Omega} \delta(\boldsymbol{x} - \boldsymbol{y})(u(\boldsymbol{y}) - v(\boldsymbol{y})) = 0, & \boldsymbol{x} \in \mathcal{M} \\ \dfrac{\partial u}{\partial \mathbf{n}}(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\mathcal{M}. \end{cases} \tag{4.10}$$

where $\partial\mathcal{M}$ is the boundary of $\mathcal{M}$ and $\mathbf{n}$ is the out normal of $\partial\mathcal{M}$. If $\mathcal{M}$ has no boundary, $\partial\mathcal{M} = \emptyset$.

The problem remaining is to solve the PDE (4.10) numerically. Notice that, we do not know the analytical form of the manifold $\mathcal{M}$. Instead, we know $\mathcal{P}(f^n)$ is a sample of the manifold $\mathcal{M}$. Then we need to solve (4.10) on this unstructured point set $\mathcal{P}(f^n)$. In this paper, we use the point integral method (PIM) [8, 14, 15] to solve (4.10) on $\mathcal{P}(f^n)$.

Algorithm 1 LDMM Algorithm - Continuous version

Require: Initial guess of the image f^0, $d^0 = 0$.
Ensure: Restored image f.

1: **while** not converge **do**
2: **while** not converge **do**
3: With fixed manifold $\mathcal{M}^n$, for $i = 1, \cdots, d$, solving

$$\alpha_i^{n+1,k+1} = \arg\min_{\alpha_i \in H^1(\mathcal{M}^n)} \|\nabla_{\mathcal{M}^n}\alpha_i\|^2_{L^2(\mathcal{M}^n)} + \mu\|\alpha_i(\mathcal{P}(f^n)) - \mathcal{P}_i(f^{n+1,k}) + d_i^k\|^2. \tag{4.6}$$

4: Update $f^{n+1,k+1}$,

$$f^{n+1,k+1} = \arg\min_{f\in\mathbb{R}^{m\times n}} \lambda\|y - \Phi f\|^2_{l^2} + \mu\|\boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f) + d^k\|^2_{\mathrm{F}} \tag{4.7}$$

5: Update d^{k+1},

$$d^{k+1} = d^k + \boldsymbol{\alpha}^{n+1,k+1}(\mathcal{P}(f^n)) - \mathcal{P}(f^{n+1,k+1}).$$

6: **end while**
7: Set $f^{n+1} = f^{n+1,k}$, $\boldsymbol{\alpha}^{n+1} = \boldsymbol{\alpha}^{n+1,k}$
8: Update $\mathcal{M}$

$$\mathcal{M}^{n+1} = \left\{(\alpha_1^{n+1}(\boldsymbol{x}), \cdots, \alpha_d^{n+1}(\boldsymbol{x})) : \boldsymbol{x} \in \mathcal{M}^n\right\}. \tag{4.8}$$

9: **end while**

4.1. Point Integral Method. The point integral method was recently proposed to solve elliptic equations over a point cloud. For the Laplace-Beltrami equation, the key observation in the point integral method is the following integral approximation.

$$\int_{\mathcal{M}} \Delta_{\mathcal{M}} u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y} \approx -\frac{1}{t}\int_{\mathcal{M}}(u(\boldsymbol{x}) - u(\boldsymbol{y}))R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y} + 2\int_{\partial\mathcal{M}}\frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}, \tag{4.11}$$

where $t > 0$ is a parameter and

$$R_t(\boldsymbol{x},\boldsymbol{y}) = C_t R\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t}\right). \tag{4.12}$$

$R : \mathbb{R}^+ \to \mathbb{R}^+$ is a positive C^2 function which is integrable over $[0, +\infty)$, and C_T is the normalizing factor

$$\bar{R}(r) = \int_r^{+\infty} R(s)\mathrm{d}s, \quad \text{and } \bar{R}_t(\boldsymbol{x},\boldsymbol{y}) = C_t\bar{R}\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t}\right) \tag{4.13}$$

We usually set $R(r) = e^{-r}$, then $\bar{R}_t(\boldsymbol{x},\boldsymbol{y}) = R_t(\boldsymbol{x},\boldsymbol{y}) = C_t\exp\left(\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t}\right)$ are Gaussians.

Next, we give a brief derivation of the integral approximation (4.11) in Euclidean space. Here we assume $\mathcal{M}$ is an open set on $\mathbb{R}^d$. For a general submanifold, the derivation follows from the same idea but is technically more involved. Interested readers are referred to [14]. Thinking of $\bar{R}_t(\boldsymbol{x},\boldsymbol{y})$ as test functions, and integrating by parts, we have

$$\begin{aligned}(4.14)\qquad &\int_{\mathcal{M}} \Delta u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}\\ =&-\int_{\mathcal{M}} \nabla u\cdot\nabla\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}+\int_{\partial\mathcal{M}}\frac{\partial u}{\partial\mathbf{n}}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}\\ =&\frac{1}{2t}\int_{\mathcal{M}}(\boldsymbol{y}-\boldsymbol{x})\cdot\nabla u(\boldsymbol{y})R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}+\int_{\partial\mathcal{M}}\frac{\partial u}{\partial\mathbf{n}}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}.\end{aligned}$$

The Taylor expansion of the function u tells us that

$$u(\boldsymbol{y})-u(\boldsymbol{x})=(\boldsymbol{y}-\boldsymbol{x})\cdot\nabla u(\boldsymbol{y})-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{x})^T\mathbf{H}_u(\boldsymbol{y})(\boldsymbol{y}-\boldsymbol{x})+O(\|\boldsymbol{y}-\boldsymbol{x}\|^3),$$

where $\mathbf{H}_u(\boldsymbol{y})$ is the Hessian matrix of u at $\boldsymbol{y}$. Note that $\int_{\mathcal{M}}\|\boldsymbol{y}-\boldsymbol{x}\|^n R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}=O(t^{n/2})$. We only need to estimate the following term.

$$\begin{aligned}(4.15)\quad &\frac{1}{4t}\int_{\mathcal{M}}(\boldsymbol{y}-\boldsymbol{x})^T\mathbf{H}_u(\boldsymbol{y})(\boldsymbol{y}-\boldsymbol{x})R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}\\ &=\frac{1}{4t}\int_{\mathcal{M}}(\boldsymbol{y}_i-\boldsymbol{x}_i)(\boldsymbol{y}_j-\boldsymbol{x}_j)\partial_{ij}u(\boldsymbol{y})R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}\\ &=-\frac{1}{2}\int_{\mathcal{M}}(\boldsymbol{y}_i-\boldsymbol{x}_i)\partial_{ij}u(\boldsymbol{y})\partial_j\left(\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\right)\mathrm{d}\boldsymbol{y}\\ &=\frac{1}{2}\int_{\mathcal{M}}\partial_j(\boldsymbol{y}_i-\boldsymbol{x}_i)\partial_{ij}u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}+\frac{1}{2}\int_{\mathcal{M}}(\boldsymbol{y}_i-\boldsymbol{x}_i)\partial_{ijj}u(\boldsymbol{y})\bar{R}_t(\boldsymbol{y},\boldsymbol{x})\mathrm{d}\boldsymbol{y}\\ &\quad-\frac{1}{2}\int_{\partial\mathcal{M}}(\boldsymbol{y}_i-\boldsymbol{x}_i)\mathbf{n}_j\partial_{ij}u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}\\ &=\frac{1}{2}\int_{\mathcal{M}}\Delta u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}-\frac{1}{2}\int_{\partial\mathcal{M}}(\boldsymbol{y}_i-\boldsymbol{x}_i)\mathbf{n}_j\partial_{ij}u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}+O(t^{1/2}).\end{aligned}$$

The second summand in the last line is $O(t^{1/2})$. Although its $L_\infty(\mathcal{M})$ norm is of constant order, its $L^2(\mathcal{M})$ norm is of the order $O(t^{1/2})$ due to the fast decay of $w_t(\boldsymbol{x},\boldsymbol{y})$. Therefore, we get Theorem 4.1 to follow from the equations (4.14) and (4.15).

THEOREM 4.1. *If $u\in C^3(\mathcal{M})$ is a function on $\mathcal{M}$, then we have for any $\boldsymbol{x}\in\mathcal{M}$,*

$$(4.16)\qquad \|r(u)\|_{L^2(\mathcal{M})}=O(t^{1/4}).$$

where

$$r(u)=\int_{\mathcal{M}}\Delta u(\boldsymbol{y})\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}+\frac{1}{t}\int_{\mathcal{M}}(u(\boldsymbol{x})-u(\boldsymbol{y}))R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}-2\int_{\partial\mathcal{M}}\frac{\partial u(\boldsymbol{y})}{\partial\mathbf{n}}R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}$$

The detailed proof can be found in [14].

Using the integral approximation (4.11), we get an integral equation to approximate the original Laplace-Beltrami equation (4.10),

$$(4.17)\qquad \int_{\mathcal{M}}(u(\boldsymbol{x})-u(\boldsymbol{y}))R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}+\mu t\sum_{\boldsymbol{y}\in\Omega}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{y})-v(\boldsymbol{y}))=0$$

This integral equation has no derivatives, and is easy to discretize over the point cloud.

4.2. Discretization. Next, we discretize the integral equation (4.17) over the point set $\mathcal{P}(f^n)$. To simplify the notation, denote the point cloud as X. Notice that the point cloud $X = \mathcal{P}(f^n)$ in the n-th iteration.

Assume that the point set $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ samples the submanifold $\mathcal{M}$ and it is uniformly distributed. The integral equation can be discretized very easily as following:

$$\frac{|\mathcal{M}|}{N}\sum_{j=1}^{N} R_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(u_i - u_j) + \mu t \sum_{j=1}^{N} \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j)(u_j - v_j) = 0 \tag{4.18}$$

where $v_j = v(\boldsymbol{x}_j)$ and $|\mathcal{M}|$ is the volume of the manifold $\mathcal{M}$.

We can rewrite (4.18) in the matrix form.

$$(\boldsymbol{L} + \bar{\mu}\bar{\boldsymbol{W}})\boldsymbol{u} = \bar{\mu}\bar{\boldsymbol{W}}\boldsymbol{v}. \tag{4.19}$$

where $\boldsymbol{v} = (v_1, \cdots, v_N)$ and $\bar{\mu} = \frac{\mu t N}{|\mathcal{M}|}$. $\boldsymbol{L}$ is a $N \times N$ matrix which is given as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \tag{4.20}$$

where $\boldsymbol{W} = (w_{ij}),\ i, j = 1, \cdots, N$ is the weight matrix and $\boldsymbol{D} = \mathrm{diag}(d_i)$ with $d_i = \sum_{j=1}^{N} w_{ij}$. $\bar{\boldsymbol{W}} = (\bar{w}_{ij}),\ i, j = 1, \cdots, N$ is also a weight matrix. From (4.18), the weight matrices are

$$w_{ij} = R_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \bar{w}_{ij} = \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{P}(f^n), \quad i, j = 1, \cdots, N. \tag{4.21}$$

Remark 4.1. *The discretization* (4.18) *is based on the assumption that the point set* $\mathcal{P}(f^n)$ *is uniformly distributed over the manifold such that the volume weight of each point is* $|\mathcal{M}|/N$. *If* $\mathcal{P}(f^n)$ *is not uniformly distributed, PIM actually solves an elliptic equation with variable coefficients [13] where the coefficients are associated with the distribution.*

Combining the point integral method within Algorithm 1, finally we get the algorithm in LDMM, Algorithm 2. In Algorithm 2, the number of split Bregman iterations is set to be 1 to simplify the computation.

5. Comparison with nonlocal methods. At first sight, LDMM is similar to nonlocal methods. But actually, they are very different. First, LDMM is based on minimizing the dimension of the patch manifold. The dimension of the patch manifold can be used as a general regularization in image processing. In this sense, LDMM is more systematic than nonlocal methods.

The other important difference is that the formulation of LDMM is continuous while the nonlocal methods use a graph based approach. In a graph based approach, a weighted graph is constructed which links different pixels x, y over the image with a weight $w(x, y)$. On this graph, the discrete gradient is defined.

$$\forall x, y, \quad \nabla_w f(x, y) = \sqrt{w(x, y)}(f(y) - f(x)). \tag{5.1}$$

Typically, in the nonlocal method, the following optimization problem is solved.

$$\min_{f \in \mathbb{R}^{m \times n}} \frac{1}{2}\|y - \Phi f\|^2 + \lambda J_{w(f)}(f). \tag{5.2}$$

Algorithm 2 LDMM Algorithm

Require: Initial guess of the image f^0, $d^0 = 0$.
Ensure: Restored image f.

1: **while** not converge **do**
2: Compute the weight matrix $W = (w_{ij})$ from $\mathcal{P}(f^n)$, where $i, j = 1, \cdots, N$ and $N = |\mathcal{P}(f^n)|$ is the total number of points in $\mathcal{P}(f^n)$,

$$w_{ij} = R_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \bar{w}_{ij} = \bar{R}_t(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad \boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{P}(f^n), \quad i, j = 1, \cdots, N.$$

And assemble the matrices $\boldsymbol{L}$, $\boldsymbol{W}$ and $\bar{\boldsymbol{W}}$ as following:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W},\ \boldsymbol{W} = (w_{i,j}),\ \bar{\boldsymbol{W}} = (\bar{w}_{i,j}),\ i, j = 1, \cdots, N.$$

3: Solve following linear systems

$$(\boldsymbol{L} + \bar{\mu}\bar{\boldsymbol{W}})\boldsymbol{U} = \bar{\mu}\bar{\boldsymbol{W}}\boldsymbol{V}.$$

where $\boldsymbol{V} = \mathcal{P}(f^n) - d^n$.
4: Update f by solving a least square problem.

$$f^{n+1} = \arg\min_{f \in \mathbb{R}^{m \times n}} \quad \lambda \|y - \Phi f\|_{l^2}^2 + \bar{\mu} \|\boldsymbol{U} - \mathcal{P}(f) + d^n\|_{\mathrm{F}}^2$$

5: Update d^n,

$$d^{n+1} = d^n + \boldsymbol{U} - \mathcal{P}(f^{n+1})$$

6: **end while**

where $J_w(f)$ is a regularization term related with the graph. Most frequently used $J_w(f)$ are the L^2 energy

$$J_w(f) = \left(\sum_{x,y} w(x,y)(f(x) - f(y))^2 \right)^{1/2} \tag{5.3}$$

and the nonlocal total variation

$$J_w(f) = \sum_x \left(\sum_y w(x,y)(f(x) - f(y))^2 \right)^{1/2} \tag{5.4}$$

The nonlocal methods provide powerful tools in image processing and were widely used in many problems. However, from the continuous point of view, the graph based approach has an intrinsic drawback.

Fig. 1 (d) shows one example of subsample image recovery computed with an L^2 energy norm. The image of Barbara (Fig. 1 (a))is subsampled. Only 10% of the pixels are retained at random (Fig. 1 (b)). It is clear that in the recovered image, some pixels are not consistent with their neighbors. From the zoomed in image, it is easy to see that these pixels are just the retained pixels. This phenomena shows that in the graph based approach, the value at the retained pixels do not spread to their

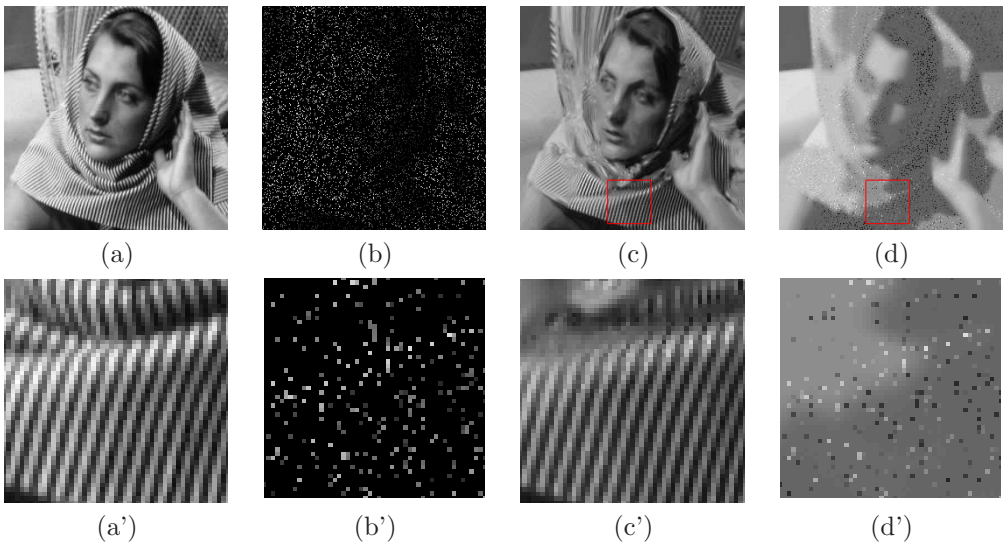

FIG. 1. *Subsampled image recovery of Barbara based on low dimensional manifold model (LDMM) and nonlocal method. (a): original image; (b): subsampled image (10% pixels are retained at random); (c): recovered image by LDMM; (d): recovered image by Graph Laplacian. The bottom row shows the zoom in image of the red box enclosed area.*

neighbours properly. Compared with the graph based approach, the result given by LDMM method is much better (Fig. 1 (c)).

This phenomena can be explained by using a simple model problem, Laplace-Beltrami equation with Dirichlet boundary condition,

$$\begin{cases} \Delta_{\mathcal{M}} u(x) = 0, & x \in \mathcal{M}, \\ u(x) = g(x), & x \in \partial\mathcal{M}. \end{cases} \tag{5.5}$$

where $\mathcal{M}$ is a smooth manifold embedded in $\mathbb{R}^d$ and $\partial\mathcal{M}$ is its boundary. Suppose the point cloud $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ samples the manifold $\mathcal{M}$ and $B \subset X$ samples the boundary $\partial\mathcal{M}$.

Using the graph based method, the solution of the Dirichlet problem (5.5) is approximately obtained by solving the following linear system

$$\begin{cases} \sum_{j=1}^{N} w(\boldsymbol{x}_i, \boldsymbol{x}_j)(u(\boldsymbol{x}_i) - u(\boldsymbol{x}_j)) = 0, & \boldsymbol{x}_i \in X \backslash B, \\ u(\boldsymbol{x}_i) = g(\boldsymbol{x}_i), & \boldsymbol{x}_i \in B. \end{cases} \tag{5.6}$$

In the point integral method, we know that the Dirichlet problem can be approximated by an integral equation

$$\begin{cases} \frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y}))R_t(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\boldsymbol{y} - 2\int_{\partial\mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}} = 0, & \boldsymbol{x} \in \mathcal{M}, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial\mathcal{M}. \end{cases} \tag{5.7}$$

Comparing these two approximations, (5.6) and (5.7), we can see that in the graph based method, the boundary term, $-2\int_{\partial\mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}} \bar{R}_t(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}}$, is dropped. However,

it is easy to check that this term is not small. Since the boundary term is dropped, the boundary condition is not enforced correctly in the graph based method.

Another difference between LDMM and the nonlocal methods is that the choice of patch is more flexible in LDMM. In nonlocal methods, for each pixel there is a patch and the patch has to be centered around this pixel. In LDMM, we only require that patches have same size and cover the whole image. This feature gives us more freedom to choose the patch.

6. Numerical results. In the numerical simulations, the weight function we used is the Gaussian weight

$$R_t(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma(\boldsymbol{x})^2}\right) \tag{6.1}$$

$\sigma(\boldsymbol{x})$ is chosen to be the distance between $\boldsymbol{x}$ and its 20th nearest neighbour, To make the weight matrix sparse, the weight is truncated to the 50 nearest neighbours.

The patch size is 10×10 in the denoising and inpainting examples and is 20×20 in the super-resolution examples.

For each point in X, the nearest neighbors are obtained by using an approximate nearest neighbor (ANN) search algorithm. We use a k-d tree approach as well as an ANN search algorithm to reduce the computational cost. The linear system in Algorithm 2 is solved by GMRES.

PSNR defined as following is used to measure the accuracy of the results

$$\mathrm{PSNR}(f, f^*) = -20\log_{10}(\|f - f^*\|/255) \tag{6.2}$$

where f^* is the ground truth.

6.1. Inpainting. In the inpainting problems, the pixels are removed from the original image and we want to recover the original image from the remaining pixels. The corresponding operator, Φ, is

$$(\Phi f)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ 0, & \boldsymbol{x} \notin \Omega, \end{cases} \tag{6.3}$$

where $\Omega \subset \{0, \cdots, m\} \times \{0, \cdots, n\}$ is the region where the image is retained. The pixels outside of Ω are removed. In our simulations, Ω is selected at random.

We assume that the original images do not have noise. In this noise free case, the parameter λ in the least-squares problem in Algorithm 2 is set to be ∞. Then, the least-squares problem can be solved as

$$f^{n+1}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ (\mathcal{P}^*\mathcal{P})^{-1}(\mathcal{P}^*(\boldsymbol{U} + d^n)), & \boldsymbol{x} \notin \Omega \end{cases} \tag{6.4}$$

where $\mathcal{P}^*$ is the adjoint operator of $\mathcal{P}$. Notice that $\mathcal{P}^*\mathcal{P}$ is a diagonal operator. So f^{n+1} can be solved explicitly without inverting a matrix.

The initial guess of f is obtained by filling the missing pixels with random numbers satisfy Gauss distribution, $N(\mu_0, \sigma_0)$, where μ_0 is the mean of Φf and σ_0 is the standard deviation of Φf. The parameter $\bar{\mu} = 0.5$.

In our simulations, the original images are subsampled. Only 10% of the pixels are retained. The retained pixels are selected at random. From these 10% subsampled images, LDMM method is employed to recover the original images. The recovery of several different images are shown in Fig. 2. In this example, we compare the

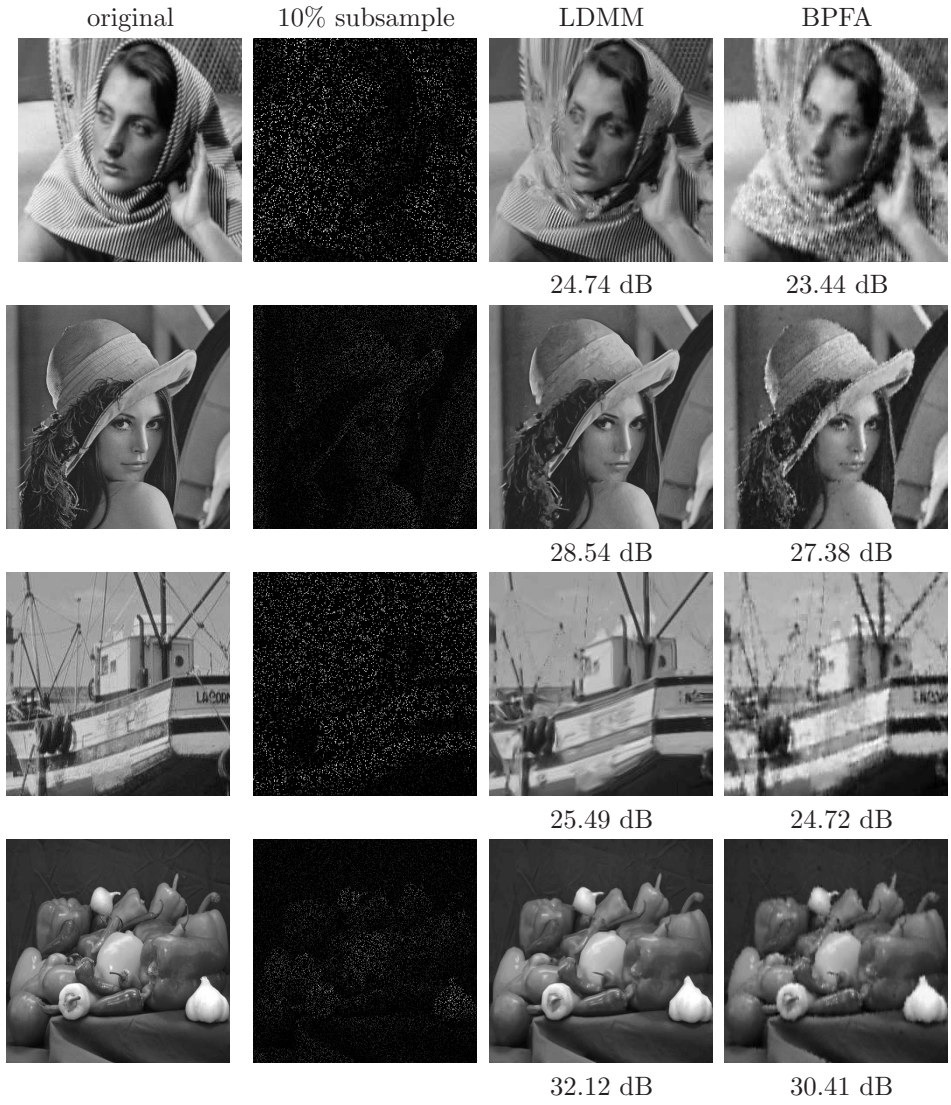

FIG. 2. *Examples of subsampled image recovery.*

performance of LDMM with BPFA [16]. As we can see in Fig. 2, the LDMM gives much better recovery. In the Barbara image, LDMM recovers the image very well both in cartoon part and the texture part.

6.2. Super-resolution. Super-resolution corresponds to the recovery of a high-definition image from a low resolution image. Usually, the low resolution image is obtained by filtering and sub-sampling from a high resolution image.

$$\Phi f = (f * h) \downarrow^k \tag{6.5}$$

where h is a low-pass filter, $\downarrow^k$ is the down-sampling operator by a factor k along each axis. Here, we consider a simple case in which the filter h is not applied, i.e.,

$$\Phi f = (f) \downarrow^k \tag{6.6}$$

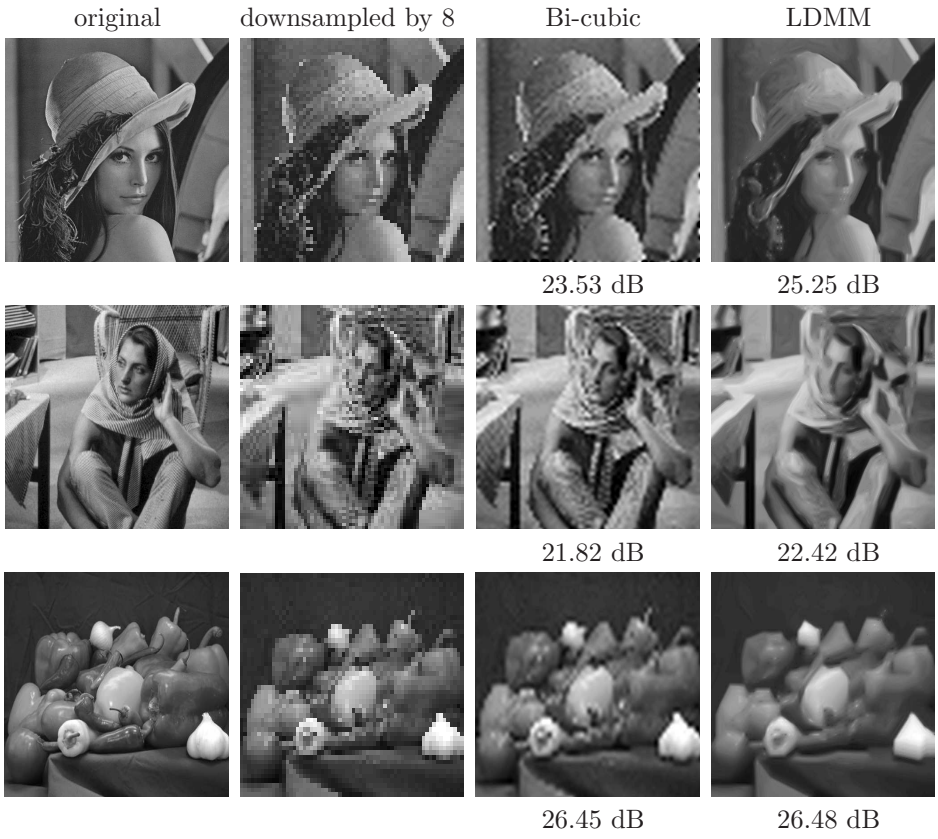

FIG. 3. *Example of image super-resolution with downsample rate $k = 8$.*

This downsample problem can be seen as a special case of subsample. However, in this problem, the pixels are retained over regular grid points which makes the recovery much more difficult than that in the random subsample problem considered in the previous example.

$$(\Phi f)(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ 0, & \boldsymbol{x} \notin \Omega, \end{cases} \tag{6.7}$$

where $\Omega = \{1, k+1, 2k+1, \cdots\} \times \{1, k+1, 2k+1, \cdots\}$.

Here, we also assume the measure is noise free and use the same formula (6.4) to update the image f. The parameter $\bar{\mu} = 0.5$ in the simulation. The initial guess is obtained by Bi-cubic interpolation.

Fig. 3 shows the results for 3 images. The downsample rate is set to be 8. In terms of PSNR, compared with Bi-cubic interpolation, the improvement in LDMM is not too much. However, the images given by LDMM are visually more pleasing than those given by Bi-cubic since the edges are reconstructed much better.

6.3. Denoising. In the denoising problem, the operator Φ is the identity operator. In this case, the least-squares problem in Algorithm 2 has closed form solution.

$$f^{n+1} = (\lambda \,\mathrm{Id} + \bar{\mu}\mathcal{P}^*\mathcal{P})^{-1} \left(\lambda y + \bar{\mu}\mathcal{P}^*(\boldsymbol{U} + d^n)\right) \tag{6.8}$$

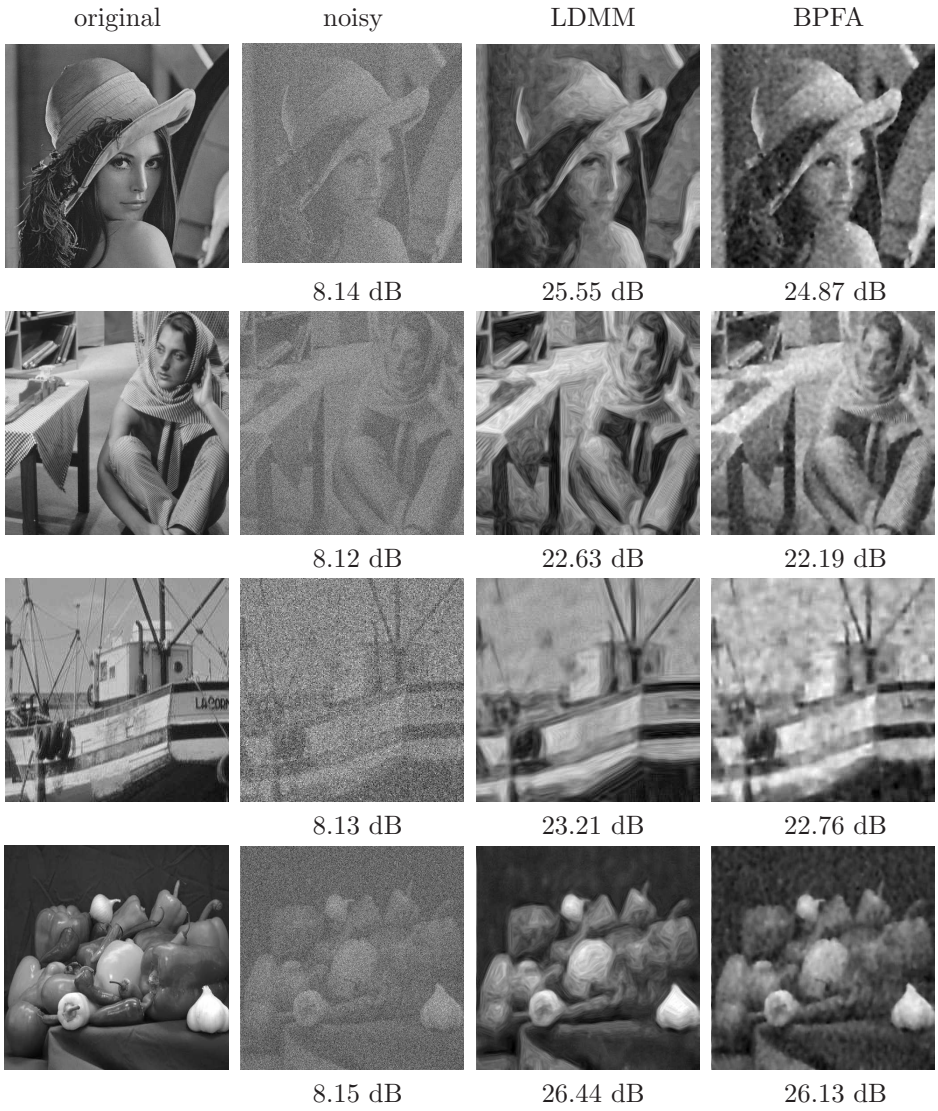

FIG. 4. *Example of image denoising with standard deviation* $\sigma = 100$.

where $\mathcal{P}^*$ is the adjoint operator of $\mathcal{P}$.

In the denoising tests, Gaussian noise is added to the original images. The standard deviation of the noise is 100. Parameter $\bar{\mu} = 0.5$ and $\lambda = 0.2$ in the simulation.

Fig. 4 shows the results obtained by LDMM. BPFA is a dictionary learning based method introduced in [16]. The results given by LDMM are a little better in terms of PSNR. However, LDMM is much better visually since edges are reconstructed better.

At the end of this section, we want to make some remarks on the computational speed of LDMM. As shown in this section, LDMM gives very good results for inpainting, super-resolution and denoising problems. On the other hand, the computational cost of LDMM is relatively high. For the example of Barbara (256×256) in inpainting

problem, LDMM needs about 18 mins while BPFA needs about 15 mins. For Barbara (512×512) in the denoising problem, LDMM spends about 9 mins (about 25 mins in BPFA). Both of the tests are run with matlab code in a laptop equiped with CPU intel i7-4900 2.8GHz. In this paper, we have not optimized the numerical method for speed. There are many possibilities to speed up. For instance, the nearest neighbors may be searched over a local window rather than over the whole image. We may also exploit the freedom to choose the index set Θ in (1.7). First, Θ is set to be a coarse set such that the number of patch is small and computation is fast. After several iteration, the result is passed to a fine grid to refine the result. Using this idea, preliminary test shows the computational time is reduced to around 5 mins for Barbara (256×256) inpainting while PSNR of the recovered image is 24.55 dB (24.74 dB in Fig. 2). These procedures will be investigated further in our future works.

7. Conclusion. In this paper, we proposed a novel low dimensional manifold model (LDMM) for image processing. In the LDMM, instead of the image itself, we study the patch manifold of the image. Many studies reveal that the patch manifold has low dimensional structure for many classes of images. In the LDMM, we just use the dimension of the patch manifold as the regularization to recover the original image from the partial information. The point integral method (PIM) also plays a very important role. It gives a correct way to solve the Laplace-Beltrami equation over the point cloud. In this paper, we show the performance of the LDMM in subsample and downsample problem. LDMM is a general model. The dimension of the patch manifold can be used as a general regularization not only in image processing problem. In some sense, LDMM is a generalization of the low rank model. We are now working on applying LDMM to matrix completion, hyperspectral image processing and large scale computational physics problems.

REFERENCES

[1] A. Buades, B. Coll, and J.-M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Model. Simul.*, 4:490–530, 2005.

[2] A. Buades, B. Coll, and J.-M. Morel. Neighborhood filters and pde's. *Numer. Math.*, 105:1–34, 2006.

[3] G. Carlsson, T. Ishkhanov, V. de Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision*, 76:1–12, 2008.

[4] G. Gilboa and S. Osher. Nonlocal linear image regularization and supervised segmentation. *Multiscale Model. Simul.*, 6:595–630, 2007.

[5] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Model. Simul.*, 7:1005–1028, 2008.

[6] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM J. Imageing Sci.*, 2:323–343, 2009.

[7] A. B. Lee, K. S. Pedersen, and D. Mumford. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54:83–103, 2003.

[8] Z. Li, Z. Shi, and J. Sun. Point integral method for solving poisson-type equations on manifolds from point clouds with convergence guarantees. *arXiv:1409.2623*.

[9] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Model. Simul.*, 4:460–489, 2005.

[10] G. Peyré. Image processing with non-local spectral bases. *Multiscale Model. Simul.*, 7:703–730, 2008.

[11] G. Peyré. Manifold models for signals and images. *Computer Vision and Image Understanding*, 113:248–260, 2009.

[12] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60:259–268, 1992.

[13] Z. Shi. Point integral method for elliptic equation with isotropic coefficients with convergence guarantee. *arXiv:1506.03606*.

[14] Z. Shi and J. Sun. Convergence of the point integral method for the poisson equation on manifolds i: the neumann boundary. *arXiv:1403.2141*.

[15] Z. Shi and J. Sun. Convergence of the point integral method for the poisson equation on manifolds ii: the dirichlet boundary. *arXiv:1312.4424*.

[16] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin. Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images. *IEEE Trans. Image Processing*, 21:130–144, 2012.