WEIGHTED GRAPH LAPLACIAN AND IMAGE INPAINTING [*]

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

Abstract. Inspired by the graph Laplacian and the point integral method, we introduce a novel weighted graph Laplacian method to compute a smooth interpolation function on a point cloud in high dimensional space. The numerical results in semi-supervised learning and image inpainting show that the weighted graph Laplacian is a reliable and efficient interpolation method. In addition, it is easy to implement and faster than graph Laplacian.

1. Introduction.

In this paper, we consider interpolation on a point cloud in high dimensional space. This is a fundamental problem in many data analysis problems and machine learning. Let $P = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n\}$ be a set of points in $\mathbb{R}^d$ and $S = \{\boldsymbol{s}_1, \cdots, \boldsymbol{s}_m\}$ be a subset of P. Let u be a function on the point set P and the value of u on $S \subset P$ is given as a function g over S, i.e. $u(\boldsymbol{s}) = g(\boldsymbol{s})$, $\forall \boldsymbol{s} \in S$. The goal of the interpolation is to find the function u on P with the given values on S.

Since the point set P is unstructured in high dimensional space, traditional interpolation methods do not apply. One model which is widely used in many applications is to minimize the following energy functional,

$$\mathcal{J}(u) = \frac{1}{2}\sum_{\boldsymbol{x},\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x}) - u(\boldsymbol{y}))^2, \tag{1.1}$$

with the constraint

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in S. \tag{1.2}$$

Here $w(\boldsymbol{x},\boldsymbol{y})$ is a given weight function. One often used weight is the Gaussian weight, $w(\boldsymbol{x},\boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|^2}{\sigma^2})$, σ is a parameter, $\|\cdot\|$ is the Euclidean norm in $\mathbb{R}^d$.

It is easy to derive the Euler-Lagrange equation of the above optimization problem, which is given as follows,

$$\begin{cases} \displaystyle\sum_{\boldsymbol{y}\in P}(w(\boldsymbol{x},\boldsymbol{y}) + w(\boldsymbol{y},\boldsymbol{x}))(u(\boldsymbol{x}) - u(\boldsymbol{y})) = 0, & \boldsymbol{x} \in P\backslash S, \\ \qquad\qquad\qquad\qquad u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in S. \end{cases} \tag{1.3}$$

This approach is the well known graph Laplacian [3, 17] which has been used widely in many problems.

Recently, it was observed that the solution given by the graph Laplacian is not continuous at the sample points, S, especially when the sample rate, $|S|/|P|$, is low [13]. Consider a simple 1D example. Let P be the union of 5000 randomly sampled points over the interval $(0, 2)$ and we label 6 points in P. Points $0, 1, 2$ are in the labeled set S and the other 3 points are selected at random. The solution given by the graph Laplacian,(1.3) is shown in Fig. 1. Clearly, the function computed by the

[*]Research supported by DOE-SC0013838 and NSF DMS-1118971. Z. Shi was partially supported by NSFC Grant 11371220, 11671005.

[†]Yau Mathematical Sciences Center, Tsinghua University, Beijing, China, 100084. *Email: zqshi@mail.tsinghua.edu.cn.*

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

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

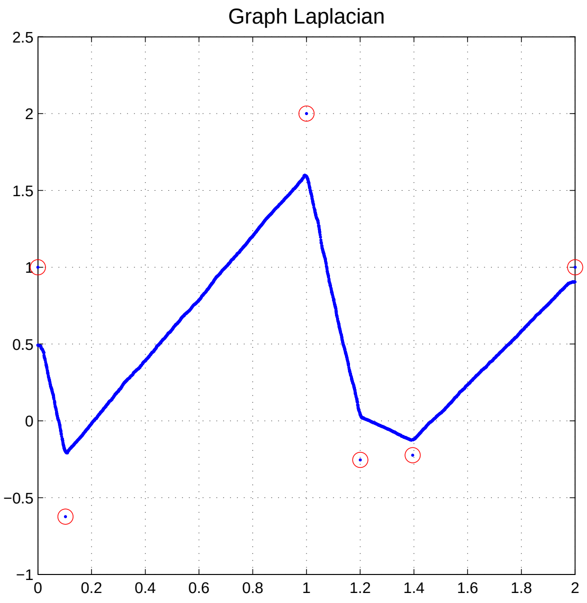

FIG. 1. *Solution given by graph Laplacian in 1D examples. Blue line: interpolation function given by graph Laplacian; red circles: given values at label set S.*

graph Laplacian is not continous at the labled points. It actually does not interpolate the given values.

It was also shown that the discontinuity is due to the fact that one important boundary term is dropped in evaluating the graph Laplacian. Consider the harmonic extension in the continuous form which is formulated as a Laplace-Beltrami equation with Dirichlet boundary condition on manifold $\mathcal{M}$,

$$\left\{ \begin{array}{rl} \Delta_{\mathcal{M}} u(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \mathcal{M}, \\ u(\boldsymbol{x}) = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial\mathcal{M}, \end{array} \right. \tag{1.4}$$

In the point integral method [6, 7, 11, 12], it is observed that the Laplace-Beltrami equation $\Delta_{\mathcal{M}} u(\boldsymbol{x}) = 0$ can be approximated by the following integral equation.

$$\frac{1}{t}\int_{\mathcal{M}} (u(\boldsymbol{x}) - u(\boldsymbol{y}))R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y} - 2\int_{\partial\mathcal{M}} \frac{\partial u(\boldsymbol{y})}{\partial \mathbf{n}}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\tau_{\boldsymbol{y}} = 0, \tag{1.5}$$

where $R_t(\boldsymbol{x},\boldsymbol{y}) = R(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t})$, $\bar{R}_t(\boldsymbol{x},\boldsymbol{y}) = \bar{R}(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{4t})$ and $\frac{\mathrm{d}}{\mathrm{d}s}\bar{R}(s) = -R(s)$. If $R(s) = \exp(-s)$, $\bar{R} = R$ and $R_t(\boldsymbol{x},\boldsymbol{y})$ becomes a Gaussian weight function. $\mathbf{n}$ is the outwards normal of the boundary $\partial\mathcal{M}$.

Comparing the above integral equation (1.5) and the equation in the graph Laplacian (1.3), we can clearly see that the boundary term in (1.5) is dropped in the graph Laplacian. However, this boundary term is not small and neglecting it causes trouble. To get a continuous interpolation, we need to include the boundary term. This is the idea of the point integral method. To deal with the boundary term in (1.5), a Robin boundary condition is used to approximate the Dirichlet boundary condition,

$$u(\boldsymbol{x}) + \mu\frac{\partial u(\boldsymbol{x})}{\partial \mathbf{n}} = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\mathcal{M}, \tag{1.6}$$

where $0 < \mu \ll 1$ is a small parameter.

$$\frac{1}{t}\int_{\mathcal{M}}(u(\boldsymbol{x})-u(\boldsymbol{y}))R_t(\boldsymbol{x},\boldsymbol{y})\mathrm{d}\boldsymbol{y}-\frac{2}{\mu}\int_{\partial\mathcal{M}}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})(g(\boldsymbol{y})-u(\boldsymbol{y}))\mathrm{d}\tau_{\boldsymbol{y}}=0.$$

The corresponding discrete equations are

$$\sum_{\boldsymbol{y}\in P}R_t(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))+\frac{2}{\mu}\sum_{\boldsymbol{y}\in S}\bar{R}_t(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{y})-g(\boldsymbol{y}))=0,\quad \boldsymbol{x}\in P, \tag{1.7}$$

The point integral method has been shown to give consistent solutions [13, 12, 6]. The interpolation algorithm based on the point integral method has been applied to image processing problems and gives promising results [9].

Equation (1.7) is not symmetric, which makes the numerical solver not very efficient. The main contribution of this paper is to propose a novel interpolation algorithm, the weighted graph Laplacian, which preserves the symmetry of the original Laplace operator. The key observation in the weighted graph Laplacian is that we need to modify the energy function (1.1) to add a weight to balance the energy on the labeled and unlabeled sets.

$$\min_u \sum_{\boldsymbol{x}\in P\backslash S}\left(\sum_{\boldsymbol{y}\in P}w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)+\frac{|P|}{|S|}\sum_{\boldsymbol{x}\in S}\left(\sum_{\boldsymbol{y}\in P}w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right),$$

with the constraint

$$u(\boldsymbol{x})=g(\boldsymbol{x}),\quad \boldsymbol{x}\in S.$$

$|P|$, $|S|$ are the number of points in P and S, respectively. When the sample rate, $|S|/|P|$, is high, the weighted graph Laplacian becomes the classical graph Laplacian. When the sample rate is low, the large weight in weighted graph Laplacian forces the solution become continuous near the labeled set.

We test the weighted graph Laplacian on MNIST dataset and image inpainting problems. The results show that the weighted graph Laplacian gives better results than the graph Laplacian, especially when the sample rate is low. The weighted graph Laplacian provides a reliable and efficient method to find reasonable interpolation on a point cloud in high dimensional space.

The rest of the paper is organized as follows. The weighted graph Laplacian is introduced in Section 2. The tests of the weighted graph Laplacian on MNIST and image inpainting are presented in Section 3 and Section 4 respectively. Some conclusions are made in Section 5.

2. Weighted Graph Laplacian.

First, we split the objective function in (1.1) to two terms, one is over the unlabeled set and the other over the labeled set.

$$\min_u \sum_{\boldsymbol{x}\in P\backslash S}\left(\sum_{\boldsymbol{y}\in P}w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)+\sum_{\boldsymbol{x}\in S}\left(\sum_{\boldsymbol{y}\in P}w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right),$$

If we substitute the optimal solution into above optimization problem, for instance the solution in the 1D example (Fig. 1), it is easy to check that the summation over the labeled set is actually pretty large due to the discontinuity on the labeled set. However, when the sample rate is low, the summation over the unlabeled set actually overwhelms the summation over the labeled set. So the continuity on the labeled set is sacrificed. One simple idea to assure the continuity on the labeled set is to put a weight ahead of the summation over the labeled set.

$$\min_u \sum_{\boldsymbol{x}\in P\backslash S}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)+\mu\sum_{\boldsymbol{x}\in S}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right),$$

This is the basic idea of our approach. Since this method is obtained by modifying the graph Laplacian to add a weight inside, we call this method the weighted graph Laplacian, WGL for short.

To balance these two terms, one natural choice of the weight μ is the inverse of the sample rate, $|P|/|S|$. Based on this observation, we get following optimization problem

(2.1)

$$\min_u \sum_{\boldsymbol{x}\in P\backslash S}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)+\frac{|P|}{|S|}\sum_{\boldsymbol{x}\in S}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right),$$

with the constraint

$$u(\boldsymbol{x})=g(\boldsymbol{x}),\quad \boldsymbol{x}\in S.$$

The optimal solution of (2.1) can be obtained by solving a linear system

$$\begin{aligned}&\sum_{\boldsymbol{y}\in P}\left(w(\boldsymbol{x},\boldsymbol{y})+w(\boldsymbol{y},\boldsymbol{x})\right)\left(u(\boldsymbol{x})-u(\boldsymbol{y})\right)\\ &\qquad +\left(\frac{|P|}{|S|}-1\right)\sum_{\boldsymbol{y}\in S} w(\boldsymbol{y},\boldsymbol{x})(u(\boldsymbol{x})-u(\boldsymbol{y}))=0,\quad \boldsymbol{x}\in P\backslash S,\\ &\qquad\qquad u(\boldsymbol{x})=g(\boldsymbol{x}),\quad \boldsymbol{x}\in S.\end{aligned}$$

This linear system is symmetric and positive definite. Comparing WGL with the graph Laplacian, in WGL, we see that a large positive term is added to the diagonal of the coefficient matrix which makes the linear system easier to solve. After solving the above linear system using some iterative method, for instance conjugate gradient, we find out that it converges faster than graph Laplacian. In our tests, the weighted graph Laplacian is about two times faster than graph Laplacian on average. In addition to the symmetry, the weighted graph Laplacian also preserves the maximum principle of the original Laplace-Beltrami operator, which is important in some applications.

Figure 2 shows the comparison between WGL and GL in the 1D example. The result of WGL perfectly interpolates the given values while GL fails.

We want to remark that there are other choices of the weight μ in WGL. $|P|/|S|$ works in many applications. Based on our experience, $|P|/|S|$ seems to be the lower bound of μ. In some applications, we may make μ larger to better fit the sample points.

We can prove the consistency of weighted graph Laplacian for the Dirichlet problem for Laplace-Beltrami equation based on a volume constraint [4, 10]. Intuitively, a large weight in WGL makes the solution close to the given value near the labeled set. It is likely to confine the solution in a volume around the labled set. Using the theory of volume constraints, we can prove the consistency of the weighted graph Laplacian.

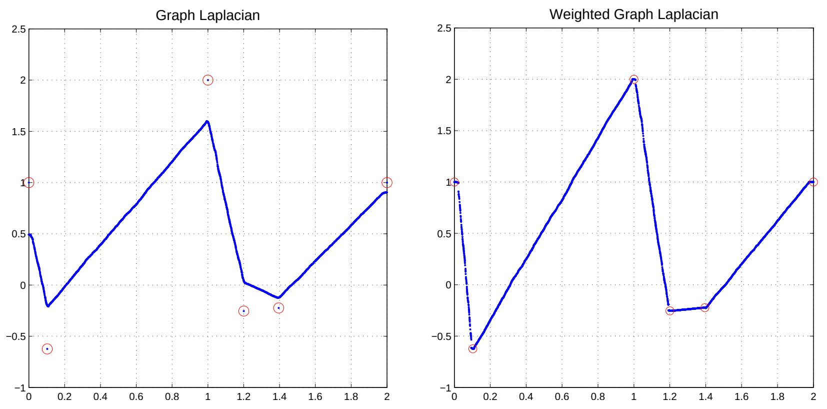

FIG. 2. *Solution given by graph Laplacian and weighted graph Laplacian in 1D examples. Blue line: interpolation function given by graph Laplacian; red circles: given values at label set S.*

The idea of weighted graph Laplacian also applies on general constraints, $Lu = b$ with L being a measurement operator and b being the observations. In this case, the labeled set S should be defined as the points involved in the measurement Lu. For instance, if L is the average operator, i.e. $\sum_{\boldsymbol{x}\in P} u(\boldsymbol{x}) = b$, then the labeled set $S = P$, although there is only one measurement.

3. Semi-supervised Learning.

In this section, we briefly describe the algorithm of semi-supervised learning based on that proposed by Zhu et al. [17]. We plug into the algorithm the aforementioned approach for weighted graph Laplacian, and apply them to the well-known MNIST dataset, and compare their performances.

Assume we are given a point set $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n\} \subset \mathbb{R}^d$, and some labels $\{1, 2, \cdots, l\}$. A subset $S \subset P$ is labeled,

$$S = \bigcup_{i=1}^{l} S_i,$$

S_i is the labeled set with label i. In a typical setting, the size of the labeled set S is much smaller than the size of the data set P. The purpose of the semi-supervised learning is to extend the label assignment to the entire P, namely, infer the labels for the unlabeled points. The algorithm is summarized in Algorithm 1.

In Algorithm 1, we use both graph Laplacian and weighted graph Laplacian to compute ϕ_i respectively. In weighted graph Laplacian, ϕ_i is obtained by solving the following linear system

$$\begin{aligned} &\sum_{\boldsymbol{y}\in P} \left(w(\boldsymbol{x},\boldsymbol{y}) + w(\boldsymbol{y},\boldsymbol{x})\right)\left(\phi_i(\boldsymbol{x}) - \phi_i(\boldsymbol{y})\right) \\ &\qquad + \left(\frac{|P|}{|S|} - 1\right)\sum_{\boldsymbol{y}\in S} w(\boldsymbol{y},\boldsymbol{x})(\phi_i(\boldsymbol{x}) - \phi_i(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in P\backslash S, \\ &\qquad \phi_i(\boldsymbol{x}) = 1, \quad \boldsymbol{x} \in S_i, \qquad \phi_i(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in S\backslash S_i. \end{aligned}$$

Algorithm 1 Semi-Supervised Learning

```
Require: A point set $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_n\} \subset \mathbb{R}^d$ and a partial labeled set $S = \cup_{i=1}^{l} S_i$.
Ensure: A complete label assignment $L : P \to \{1, 2, \cdots, l\}$
  for $i = 1 : l$ do
    Compute $\phi_i$ on $P$, with the constraint
        $\phi_i(\boldsymbol{x}) = 1,\ \boldsymbol{x} \in S_i, \quad \phi_i(\boldsymbol{x}) = 0,\ \boldsymbol{x} \in S \backslash S_i,$
  end for
  for $(\boldsymbol{x} \in P \backslash S)$ do
    Lable $\boldsymbol{x}$ as following
        $L(\boldsymbol{x}) = k, \quad \text{where} \quad k = \arg\max_{1 \le i \le l} \phi_i(\boldsymbol{x}).$
  end for
```

In graph Laplacian, we need to solve

$$\sum_{\boldsymbol{y} \in P} (w(\boldsymbol{x}, \boldsymbol{y}) + w(\boldsymbol{y}, \boldsymbol{x}))\,(\phi_i(\boldsymbol{x}) - \phi_i(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in P \backslash S,$$

$$\phi_i(\boldsymbol{x}) = 1, \quad \boldsymbol{x} \in S_i, \qquad \phi_i(\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in S \backslash S_i.$$

We test Algorithm 1 on MNIST dataset of handwritten digits [2]. MNIST dataset contains $70,000$ 28×28 gray scale digit images. We view digits $0 \sim 9$ as ten classes. Each image can be seen as a point in 784-dimensional Euclidean space. The weight

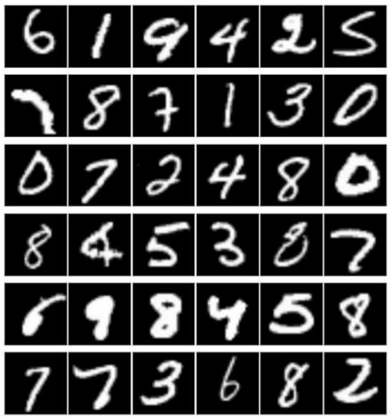

Fig. 3. *Some images in MNIST dataset.*

function $w(\boldsymbol{x}, \boldsymbol{y})$ is constructed as following

$$w(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{\sigma(\boldsymbol{x})^2}\right) \tag{3.1}$$

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

100/70,000		70/70,000		50/70,000	
WGL	GL	WGL	GL	WGL	GL
91.83%	74.56%	84.31%	29.67%	79.20%	33.93%
89.20%	58.16%	87.74%	42.29%	78.66%	21.12%
93.13%	62.98%	83.19%	37.02%	71.29%	24.50%
90.43%	41.27%	91.08%	12.57%	71.92%	29.79%
92.27%	44.57%	84.74%	35.15%	80.92%	37.50%

Table 1
Accuracy of weighted graph Laplacian and graph Laplacian in the test of MNIST.

In our test, we label 100, 70 and 50 images respectively. The labeled images are selected at random in 70,000 images. For each case, we do 5 independent tests and the results are shown in Table 1. It is quite clear that WGL has a better performance than GL. The average accuracy of WGL is much higher than that of GL. In addition, WGL is more stable than GL. The fluctuation of GL in different tests is much higher. Due to the inconsistency, in GL, the values in the labeled points are not well spread on to the unlabeled points. On many unlabeled points, the function ϕ_i is actually close to 1/2. This makes the classification sensitive to the distribution of the labeled points.

As for the computational time, in our tests, WGL takes about half the time of GL on average, 15s vs 29s (not including the time to construct the weight), with matlab code in a laptop equiped with CPU intel i7-4900 2.8GHz. In WGL, a positive term is added to the diagonal of the coefficient matrix which makes conjugate gradient converge faster.

As a remark, in this paper, we do not intend to give a state-of-the-art method in semi-supervised learning. We just use this example to test the weighted graph Laplacian.

4. Image Inpainting.

In this section, we apply the weighted graph Laplacian to the reconstruction of subsampled images. To apply the weighted graph Laplacian, first, we construct a point cloud from a given image by taking patches. We consider a discrete image $f \in \mathbb{R}^{m\times n}$. For any $(i,j) \in \{1,2,\dots,m\} \times \{1,2,\dots,n\}$, we define a patch $p_{ij}(f)$ as a 2D piece of size $s_1 \times s_2$ of the original image f, and the pixel (i,j) is the center of the rectangle of size $s_1 \times s_2$. The patch set $\mathcal{P}(f)$ is defined as the collection of all patches:

$$\mathcal{P}(f) = \{p_{ij}(f) : (i,j) \in \{1,2,\dots,m\} \times \{1,2,\dots,n\}\} \subset \mathbb{R}^d, \quad d = s_1 \times s_2.$$

For a given image f, the patch set $\mathcal{P}(f)$ gives a point cloud in $\mathbb{R}^d$ with $d = s_1 \times s_2$. We also define a function u on $\mathcal{P}(f)$. At each patch, the value of u is defined to be the intensity of image f at the central pixel of the patch, i.e.

$$u(p_{ij}(f)) = f(i,j),$$

where $f(i,j)$ is the intensity of image f at pixel (i,j).

Now, we subsample the image f in the subsample domain $\Omega \subset \{(i,j) : 1 \le i \le m,\ 1 \le j \le n\}$. The problem is to recover the original image f from the subsamples $f|_\Omega$. This problem can be transfered to interpolation of function u in the patch set $\mathcal{P}(f)$ with u is given in $S \subset \mathcal{P}(f)$, $S = \{p_{ij}(f) : (i,j) \in \Omega\}$. Notice that the patch set $\mathcal{P}(f)$ is not known, we need to update the patch set iteratively from the recovered

Algorithm 2 Subsample image restoration

```
Require: A subsample image $f|_\Omega$.
Ensure: A recovered image $u$.
  Generate initial image $u^0$.
  while not converge do
    1. Generate patch set $\mathcal{P}(u^n)$ from current image $u^n$ and get corresponding labeled set $S^n \subset \mathcal{P}(u^n)$.
    2. Compute the weight function $w^n(\boldsymbol{x}, \boldsymbol{y})$ for $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{P}(u^n)$.
    3. Update the image by computing $u^{n+1}$ on $\mathcal{P}(u^n)$, with the constraint
           $u^{n+1}(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in S^n.$
    4. $n \leftarrow n + 1$.
  end while
  $u = u^n$.
```

image. Summarizing this idea, we get an algorithm to reconstruct the subsampled image which is stated in Algorithm 2.

There are different methods to compute u^{n+1} on $\mathcal{P}(u^n)$ in Algorithm 2. In this paper, we use weighted graph Laplacian and graph Laplacian to compute u^{n+1}. In the weighted graph Laplacian, we need to solve the following linear system

$$\begin{aligned}
&\sum_{\boldsymbol{y}\in\mathcal{P}(u^n)} (w^n(\boldsymbol{x}, \boldsymbol{y}) + w^n(\boldsymbol{y}, \boldsymbol{x})) \left(u^{n+1}(\boldsymbol{x}) - u^{n+1}(\boldsymbol{y})\right) \\
&\qquad + \left(\frac{mn}{|\Omega|} - 1\right) \sum_{\boldsymbol{y}\in S^n} w^n(\boldsymbol{y}, \boldsymbol{x})(u^{n+1}(\boldsymbol{x}) - u^{n+1}(\boldsymbol{y})) = 0, \quad \boldsymbol{x} \in \mathcal{P}(u^n)\backslash S^n, \\
&\qquad\qquad u^{n+1}(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in S^n.
\end{aligned}$$

While in the graph Laplacian, we solve the other linear system,

$$\begin{aligned}
\sum_{\boldsymbol{y}\in\mathcal{P}(u^n)} (w^n(\boldsymbol{x}, \boldsymbol{y}) + w^n(\boldsymbol{y}, \boldsymbol{x})) \left(u^{n+1}(\boldsymbol{x}) - u^{n+1}(\boldsymbol{y})\right) = 0, &\quad \boldsymbol{x} \in \mathcal{P}(u^n)\backslash S^n, \\
u^{n+1}(\boldsymbol{x}) = f(\boldsymbol{x}), &\quad \boldsymbol{x} \in S^n.
\end{aligned}$$

Actually, Algorithm 2 with graph Laplacian is just a nonlocal method in image processing[1, 5]. In nonlocal methods, people try to minimize energy functions such as

$$\min_u \quad \sum_{i=1}^{m}\sum_{j=1}^{n} |\nabla_w u(i, j)|^2,$$

with the constraint

$$u(i, j) = f(i, j), \quad (i, j) \in \Omega.$$

$\nabla_w u(i, j)$ is the nonlocal gradient which is defined as

$$\nabla_w u(i, j) = \sqrt{w(i, j; i', j')}(u(i', j') - u(i, j)), \quad 1 \leq i, i' \leq m,\ 1 \leq j, j' \leq n.$$

$w(i,j;i',j')$ is the weight from pixel (i,j) to pixel (i',j'),

$$w(i,j;i',j') = \exp\left(\frac{d(f(i,j),f(i',j'))}{\sigma^2}\right)$$

$d(f(i,j),f(i',j'))$ is the patch distance in image f,

$$d(f(i,j),f(i',j')) = \sum_{k=-h_1}^{h_1}\sum_{l=-h_2}^{h_2}\chi(k,l)|f(i+k,j+l)-f(i'+k,j'+l)|^2$$

χ is often chosen to be 1 or a Gaussian, h_1, h_2 are half sizes of the patch. It is easy to check that the above nonlocal method is the same as solving the graph Laplacian on the patch set. If the weight w is updated iteratively, we get Algorithm 2 with the graph Laplacian. Next, we will show that this method has inconsistent results, and we can use the weighted graph Laplacian to address this issue.

In our calculations below, we take the weight $w(\boldsymbol{x},\boldsymbol{y})$ as following:

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

$\sigma(\boldsymbol{x})$ is chosen to be the distance between $\boldsymbol{x}$ and its 20th nearest neighbor, To make the weight matrix sparse, the weight is truncated to the 50 nearest neighbors. The patch size is 11×11. For each patch, 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 weighted graph Laplacian and graph Laplacian is solved by the conjugate gradient method.

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

$$\text{PSNR}(f,f^*) = -20\log_{10}(\|f-f^*\|/255) \tag{4.1}$$

where f^* is the ground truth.

First, we run a simple test to see the performance of weighted graph Laplacian and graph Laplacian. In this test, the patch set is constructed using the original image, Figure 4(a). The original image is subsampled at random, only keeping 1% pixels. Since the patch set is exact, we do not update the patch set and only run WGL and GL once to get the recovered image.

The results of WGL and GL are shown in Figure 4(c),(d) respectively. Obviously, the result of WGL is much better. To have a closer look at of the recovery, Figure 5 shows the zoomed in image enclosed by the boxes in Figure 4(a). In Figure 5(d), there are many pixels which are not consistent with their neighbors. Compared with the subsample image 5(b), it is easy to check that these pixels are actually the retained pixels. The reason is that in graph Laplacian a non-negligible boundary term is dropped [13, 12]. On the contrary, in the result of WGL, the inconsistency disappears and the resultant recovery is much better and smoother as shown in Figure 4(d) and 5(d).

At the end of this section, we apply Algorithm 2 to recover the subsampled image. In this test, we modify the patch to add the local coordinate

$$\bar{p}_{ij}(f) = [p_{ij}(f), \lambda_1 i, \lambda_2 j]$$

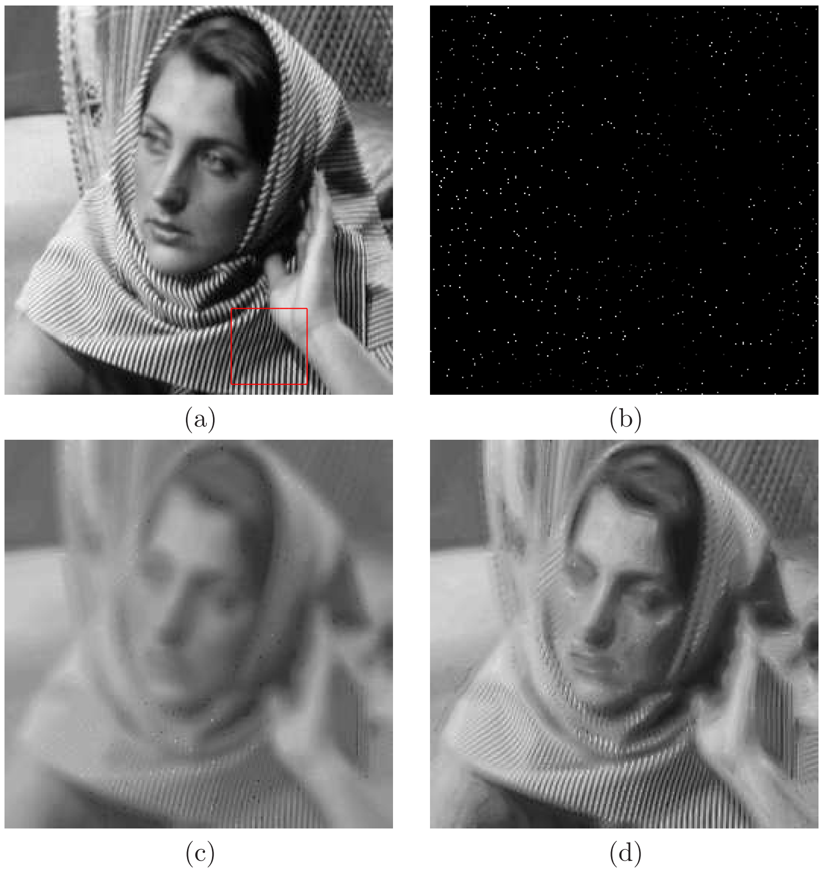

FIG. 4. *Test of image restoration on image of Barbara. (a): original image; (b): 1% subsample; (c): result of GL; (d): result of WGL.*

with

$$\lambda_1 = \frac{3\|(f|_\Omega)\|_\infty}{m}, \quad \lambda_2 = \frac{3\|(f|_\Omega)\|_\infty}{n}.$$

This semi-local patch could accelerate the iteration and give better reconstruction. The number of iterations in our computation is fixed to be 10.

The initial image is obtained by filling the missing pixels with random numbers which satisfy a Gaussian distribution, where μ_0 is the mean of $f|_\Omega$ and σ_0 is the standard deviation of $f|_\Omega$.

The results are shown in Figure 6. As we can see, WGL gives much better results than GL both visually and numerically in PSNR. The results are comparable with those in LDMM [9] while WGL is much faster. For the image of Barbara (256×256), WGL needs about 1 minutes and LDMM needs about 20 minutes in a laptop equiped with CPU intel i7-4900 2.8GHz with matlab code. In WGL and GL, the weight update is the most computationally expensive part in the algorithm by taking more than 90% of the entire computational time. This part is the same in WGL and GL. So the total time of WGL and GL are almost same, although the time of solving the linear system is about half in WGL, 1.7s vs 3.1s.

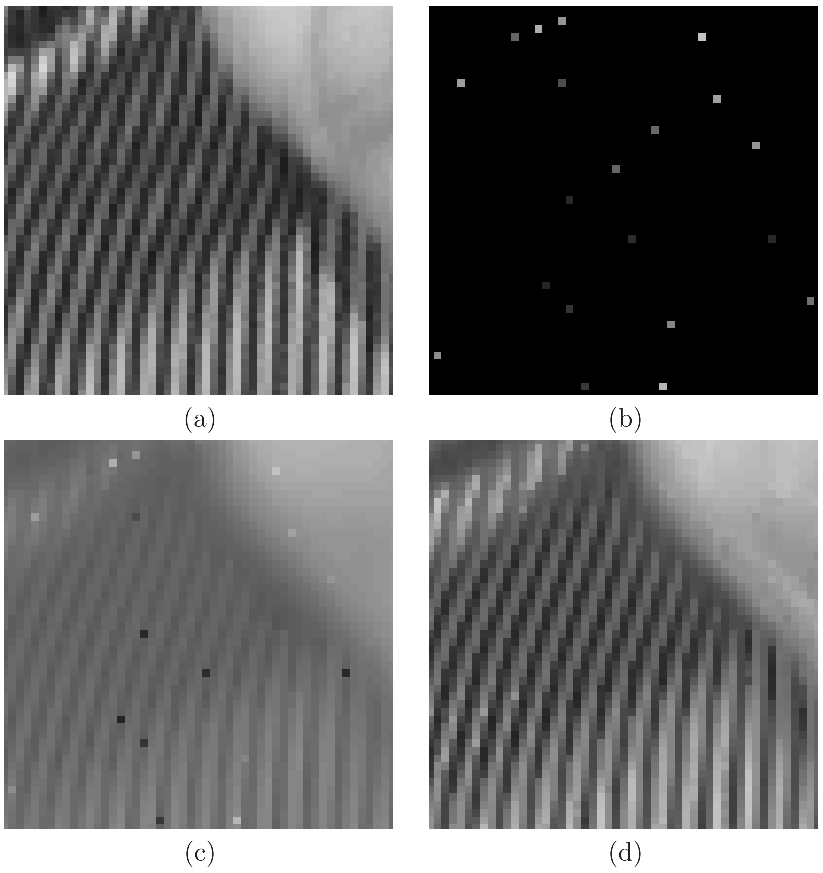

FIG. 5. *Zoomed in image in the test of image restoration on image of Barbara. (a): original image; (b): 1% subsample; (c): result of GL; (d): result of WGL.*

5. Conclusion and Future Work. In this paper, we introduce a novel weighted graph Laplacian method. The numerical results show that weighted graph Laplacian provides a reliable and efficient method to find reasonable interpolation on a point cloud in high dimensional space.

On the other hand, it was found that with extremely low sample rate, formulation of the harmonic extension may fail [8, 15]. In this case, we are considering using the ∞-Laplacian to compute the interpolation on point cloud, i.e. solving the following optimization problem

$$\min_{u}\left(\max_{\boldsymbol{x}\in P}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)^{1/2}\right),$$

with the constraint

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x}\in S.$$

This work will be reported in our subsequent paper soon.

Another interesting problem is the semi-supervised learning studied in Section 3. In semi-supervised learning, ideally, the functions, ϕ_i, should be either 0 or 1, so

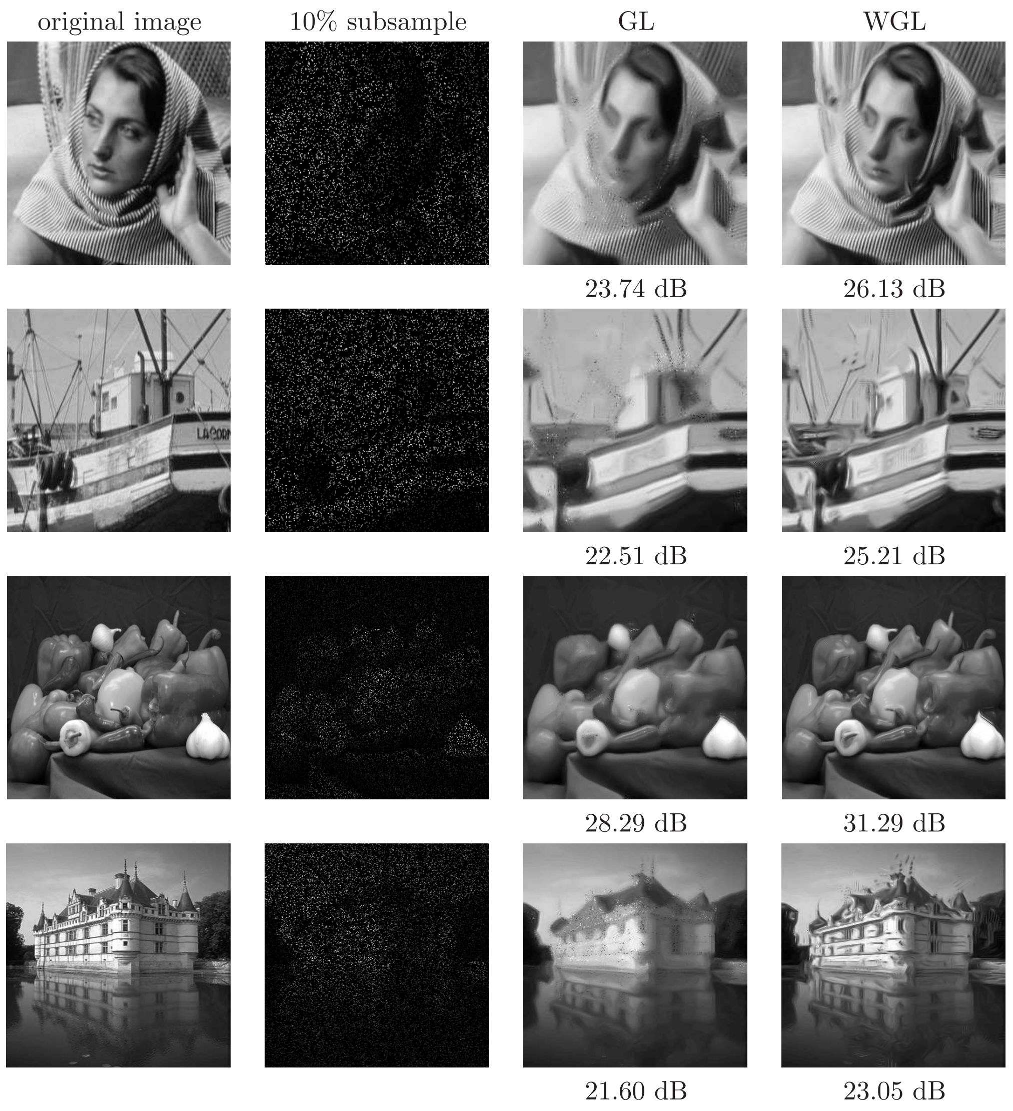

FIG. 6. *Results of subsample image restoration.*

they are piecewise constant. In this sense, minimizing the total variation should give better results [5, 14, 16]. Based on the weighted graph Laplacian, we should also add a weight to correctly enforce the constraints on the labeled points. This idea implies the following weighted nonlocal TV method,

$$\min_u \sum_{\boldsymbol{x}\in P\setminus S}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)^{1/2} + \frac{|P|}{|S|}\sum_{\boldsymbol{x}\in S}\left(\sum_{\boldsymbol{y}\in P} w(\boldsymbol{x},\boldsymbol{y})(u(\boldsymbol{x})-u(\boldsymbol{y}))^2\right)^{1/2},$$

with the constraint

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x}\in S.$$

This seems to be a better approach than the weighted graph Laplacian in the semi-supervised learning. We will explore this approach and report its performance in our future work.

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] C. J. Burges, Y. LeCun, and C. Cortes. Mnist database.

[3] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[4] Q. Du, M. Gunzburger, R. B. Lehoucq, and K. Zhou. Analysis and approximation of nonlocal diffusion problems with volume constraints. *SIAM Review*, 54:667–696, 2012.

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

[6] Z. Li and Z. Shi. A convergent point integral method for isotropic elliptic equations on point cloud. *SIAM: Multiscale Modeling Simulation*, 14:874–905, 2016.

[7] 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*.

[8] B. Nadler, N. Srebro, and X. Zhou. Semi-supervised learning with the graph laplacian: The limit of infinite unlabelled data. *NIPS*, 2009.

[9] S. Osher, Z. Shi, and W. Zhu. Low dimensional manifold model for image processing. *Technical Report, CAM report 16-04, UCLA*, 2016.

[10] Z. Shi. Enforce the dirichlet boundary condition by volume constraint in point integral method. *arXiv:1506.02343*.

[11] Z. Shi and J. Sun. Convergence of the point integral method for poisson equation on point cloud. *arXiv:1403.2141*.

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

[13] Z. Shi, J. Sun, and M. Tian. Harmonic extension on point cloud. *arXiv:1509.06458*.

[14] K. Yin, X.-C. Tai, and S. Osher. An effective region force for some variational models for learning and clustering. *Technical Report, CAM report 16-18, UCLA*, 2016.

[15] X. Zhou and M. Belkin. Semi-supervised learning by higher order regularization. *NIPS*, 2011.

[16] W. Zhu, V. Chayes, A. Tiard, S. Sanchez, D. Dahlberg, D. Kuang, A. Bertozzi, S. Osher, and D. Zosso. Nonlocal total variation with primal dual algorithm and stable simplex clustering in unspervised hyperspectral imagery analysis. *Technical Report, CAM report 15-44, UCLA*, 2015.

[17] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Machine Learning, Proceedings of the Twentieth International Conference ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 912–919, 2003.