

Stat 992: Lecture 27

Laplacian in regular grid.

Moo K. Chung mchung@stat.wisc.edu

April 2, 2004

The lecture is based on **Problem 27**.

1. *4 neighbor Laplacian*. Consider a regular grid system $\delta l \mathbb{Z}^2$. It was shown that the 4 neighbor Laplacian weights are given as

$$(w_{ij}) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Consider continues function f evaluated in the neighborhood of (x_0, y_0) . The Laplacian is

$$\Delta f = \frac{1}{\delta l^2} \sum_{i,j=-1,0,1} w_{ij} f(x_i, y_i)$$

and the iterative scheme is

$$f(x_0, y_0) \leftarrow f(x_0, y_0) + \lambda \sum_{i,j} w_{ij} f(x_i, y_i)$$

where $\lambda = \delta t / \delta l^2$ is the diffusion coefficient. This can be written as

$$f(x_0, y_0) \leftarrow \sum_{i,j} W_{ij} f(x_i, y_i)$$

where

$$(W_{ij}) = \begin{pmatrix} 0 & \lambda & 0 \\ \lambda & 1 - 4\lambda & \lambda \\ 0 & \lambda & 0 \end{pmatrix}.$$

Note that $\sum_{i,j} W_{ij} = 1$.

2. *8 neighbor Laplacian*. Since $K_{\sqrt{2\delta t}} * f$ is the solution of the diffusion equation after time δt , we let

$$f(x_0, y_0) + \lambda \sum_{i,j} w_{ij} f(x_i, y_i) \approx K_{\sqrt{2\delta t}} * f(x_0, y_0).$$

Similar formulation has been used to estimates the weights in terms of exponential functions in Belkin, M. and Niyogi, P., Laplacian eigenmaps for dimensionality reduction and data representation (2002).

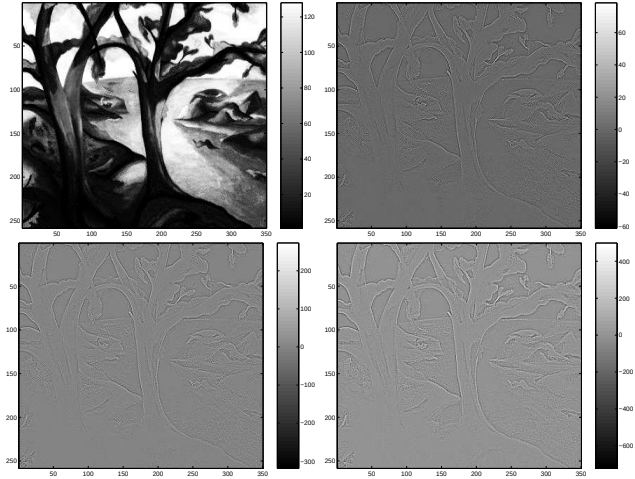


Figure 1: Top left: original image, Top right: 4 neighbor Laplacian of the image based on built in MATLAB function. Bottom left: 4 neighbor Laplacian of the image based on Laplacian4. Bottom right: 8 neighbor Laplacian of the image based on Laplacian8.

Due to symmetry, we only need to evaluate the kernel at $(0, 0), (\delta l, 0), (\delta l, \delta l)$. $K(0, 0) \propto \frac{1}{\delta t}$, $K(\delta l, 0) \propto \frac{1}{\delta t} e^{-\lambda/4} \approx \frac{1}{\delta t} (1 - \frac{\lambda}{4})$ and $K(\delta l, \delta l) \propto \frac{1}{\delta t} e^{-\lambda/2} \approx \frac{1}{\delta t} (1 - \frac{\lambda}{2})$. Since δl is small, the integral can be estimated by the finite sum so that weights are proportional to $1, 1 - \lambda/4, 1 - \lambda/2$. Now the kernel weights are given by

$$(W_{ij}) = \frac{1}{9 - 3\lambda} \begin{pmatrix} 1 - \frac{\lambda}{2} & 1 - \frac{\lambda}{4} & 1 - \frac{\lambda}{2} \\ 1 - \frac{\lambda}{4} & 1 & 1 - \frac{\lambda}{4} \\ 1 - \frac{\lambda}{2} & 1 - \frac{\lambda}{4} & 1 - \frac{\lambda}{2} \end{pmatrix}.$$

The weights are normalized so that $\sum_{i,j} W_{ij} = 1$. Then the 8 neighbor Laplacian weight is

$$(w_{ij}) = \frac{1}{9 - 3\lambda} \begin{pmatrix} 1 - \frac{\lambda}{2} & 1 - \frac{\lambda}{4} & 1 - \frac{\lambda}{2} \\ 1 - \frac{\lambda}{4} & 3\lambda - 8 & 1 - \frac{\lambda}{4} \\ 1 - \frac{\lambda}{2} & 1 - \frac{\lambda}{4} & 1 - \frac{\lambda}{2} \end{pmatrix}.$$

Since λ is very small, we have 8 neighbor Laplacian

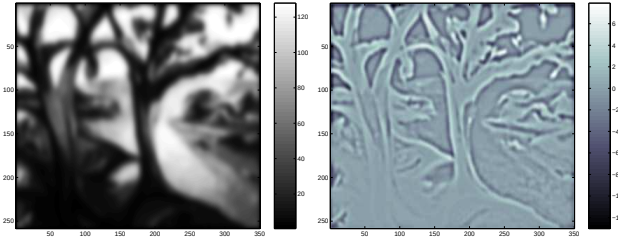


Figure 2: Left: smoothed image. Right: Laplacian of smoothed image. Comparing with the previous figures, it gives more crisp edges due to the suppression of noise.

weights

$$(w_{ij}) = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

```
load trees
imagesc(X);colormap('bone');colorbar
L = del2(X)%built-in Laplacian.
```

```
Laplacian4=[0 1 0
            1 -4 1
            0 1 0]
L4X=conv2(X,Laplacian4,'same');
```

```
Laplacian8=[1 1 1
            1 -8 1
            1 1 1]
L8X=conv2(X,Laplacian8,'same');
```

Problem 40. Based on the above 4 neighbor liberalization of the above Laplacian, find function f that minimize

$$\sum_{ij} (Y(p_{ij}) - f(p_{ij}))^2 + \lambda \int_{\Omega} \Delta f(p) f(p) d\mu(p).$$

The integral can be approximated as $\sum_{ij} \Delta f(p_{ij}) f(p_{ij})$. f gives a smoothing of data Y defined on Ω . Apply it to the sagittal image Ω and smooth the image intensity $Y(p_{ij})$ this way. Hint: Read Belkin and Niyogi's Semi-Supervised Learning on Riemannian Manifolds (2003). I think Tim Ramsay (2000) has done something similar and called it Laplacian smoothing.

3. *Edge detection.* In the place where the image intensity changes rapidly, the Laplacian will have extreme values. This is how the Laplacian can be used to detect edges in images without much efforts; however the Laplacian is not well defined in images due to the discreteness of image intensity.

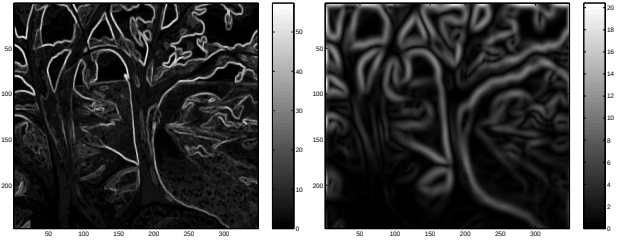


Figure 3: Left: local variance map. Right: local variance map of smoothed image. It is the variance map of Figure 2 left image.

To make it more smooth, kernel smoothing is usually performed before computing the Laplacian, i.e. $\Delta K_{\sigma} * Y$.

```
Gaussian=[0 1/2 0
          1/2 1 1/2
          0 1/2 0]/3
GX=X;
for i=1:100
    GX=conv2(GX,Gaussian,'same');
end;
GL4X=conv2(GX,Laplacian4,'same');
```

An alternate much simpler way to detect edge may be to use local variance map. For image intensity Y , let $\text{Var}_d(Y)$ to be the variance map of Y where the sample variance is computed locally in square of size $d \times d$. In the regions of homogenous image intensity, intensity variance doesn't change much while in the region of rapidly changing image intensity, $\text{Var}_d(Y)$ will have large values. So the local intensity variance can be used to extract edge information. In practice, $\text{Var}_d(Y)$ is not a continuous function. To make it more smooth, possibly for modeling the covariance function, we smooth with Gaussian kernel before the variance computation, i.e. $\text{Var}_d(K_{\sigma} * Y)$.

```
for i=2:247
    for j=2:348
        SX(i,j)=std(reshape(X((i-1)...
                               : (i+1), (j-1):(j+1)), 9, 1));
    end
end;
```

Problem 41. Estimate the covariance function $R(t, s)$ of the smoothed tree image (Figure 2 Left). $z = R(t, s)$ gives a 2D surface with respect to variables t and s . Plot your covariance function surface. You may need to use `mesh` in MATLAB.