Mathematical Problems in Image Processing

Total Variation Denoising-ROF model and Chambolle's Projection Algorithm

主讲人 邱欣欣
幻灯片制作 邱欣欣

中国海洋大学 信息科学与工程学院

2013 年 12 月 6 日

Total Variation Denoising

Contents

Introduction

- A common model is $f = Ru + \eta$. The operator R represents a linear operator and η represents the additive noise.
- The first idea to recover the image u by minimization of energy was proposed by Tikhonov and Arsenin.

$$F(u) = \int_\Omega |\nabla u|^2 + \lambda \int_\Omega |f - Ru|^2 \mathrm{d}x$$

Introduction

- Rudin, Osher, and Fatemi considered minimizing the following functional
$$J(u) = \int_\Omega |\nabla u| + \lambda \int_\Omega (f - u)^2 \mathrm{d}x$$
- Aubert and Vese studied minimization of the following functional, which was originally proposed by D.Geman and S.Geman.

$$E(u) = \int_\Omega \phi(|\nabla u|) + \lambda \int_\Omega (f - Ru)^2 \mathrm{d}x$$

Contents

Rudin-Osher-Fatemi model (ROF model)

DEFINITION 1

The total variation of an image is defined by duality: for $u \in L^1_{loc}(\Omega)$ it is given by

$$J(u) = \sup\left\{ -\int_\Omega u \operatorname{div}\phi \, dx : \phi \in C_c^\infty(\Omega;\mathbb{R}^N), |\phi(x)| \leqslant 1 \ \forall x \in \Omega \right\}$$

J is finite if and only if the distributional derivative Du of u is a finite Radon measure in Ω, in which case we have $J(u) = |Du|(\Omega)$. If u has a gradient $\nabla u \in L^1(\Omega;\mathbb{R}^2)$, then $J(u) = \int_\Omega |\nabla u(x)| \mathrm{d}x$.

Rudin-Osher-Fatemi model (ROF model)

- $$f = u + \eta$$

  The model to consider the minimization of the total variation was introduced by Rudin, Osher and Fatemi.

- $$J(u) = \int_\Omega |\nabla u| + \lambda \int_\Omega (f - u)^2 \mathrm{d}x$$

  It is based on the principle that signals with excessive and possibly spurious detail have high total variation, that is, the integral of the absolute gradient of the signal is high.

Total variation denoising-ROF model

- The Euler Lagrange differential equation for minimization of $J(u)$ is as follows

$$-\mathrm{div}(\frac{\nabla u}{|\nabla u|}) + 2\lambda(u - f) = 0$$

$$u = f + \frac{1}{2\lambda}\mathrm{div}(\frac{\nabla u}{|\nabla u|}) \text{ in } \Omega$$

The Neumann boundary condition is

$$\frac{\partial u}{\partial v} = 0 \text{ on the boundary } \partial\Omega$$

$$v = f - u$$

Total variation denoising-ROF model

- The functional is replaced by its regularized form

$$\mathcal{F}^{\varepsilon}(u) = \int_{\Omega} \sqrt{\varepsilon^2 + |\nabla u|^2} + \lambda \int_{\Omega} (f - u)^2 \mathrm{d}x$$

$$u = f + \frac{1}{2\lambda} \operatorname{div}\left(\frac{\nabla u}{\sqrt{\varepsilon^2 + |\nabla u|^2}}\right) \text{ in } \Omega$$

$$\frac{\partial u}{\partial v} = 0 \text{ on the boundary } \partial\Omega$$

Total variation denoising-ROF model

- The region Ω is covered with computational grid $(x_i, y_j) = (ih, jh)$ where h is a cell size. Let $D_+ = D_+(h), D_- = D_-(h)$, and $D_0 := (D_+ + D_-)/2$ denote the usual forward, backward, and centered divided difference.
- Thus, $D_{+x}u_{i,j} = (u_{i+1,j} - u_{i,j})/h$, $D_{-x}u_{i,j} = (u_{i,j} - u_{i-1,j})/h$, $D_{+y}u_{i,j} = (u_{i,j+1} - u_{i,j})/h$, $D_{-y}u_{i,j} = (u_{i,j} - u_{i,j-1})/h$, $D_{0x}u_{i,j} = (u_{i+1,j} - u_{i-1,j})/2h$ and $D_{0y}u_{i,j} = (u_{i,j+1} - u_{i,j-1})/2h$.

Total variation denoising-ROF model

A discrete form of the Euler-Lagrange equation is:

$$
\begin{aligned}
u_{i,j} &= f_{i,j} + \frac{1}{2\lambda} D_{-x}\left[\frac{D_{+x}u_{i,j}}{\sqrt{\varepsilon^2 + (D_{+x}u_{i,j})^2 + (D_{0y}u_{i,j})^2}}\right] \\
&\quad + \frac{1}{2\lambda} D_{-y}\left[\frac{D_{+y}u_{i,j}}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j})^2 + (D_{+y}u_{i,j})^2}}\right] \\
&= f_{i,j} + \frac{1}{2\lambda h^2}\left[\frac{u_{i+1,j} - u_{i,j}}{\sqrt{\varepsilon^2 + (D_{+x}u_{i,j})^2 + (D_{0y}u_{i,j})^2}}\right. \\
&\quad \left. - \frac{u_{i,j} - u_{i-1,j}}{\sqrt{\varepsilon^2 + (D_{-x}u_{i,j})^2 + (D_{0y}u_{i-1,j})^2}}\right] \\
&\quad + \frac{1}{2\lambda h^2}\left[\frac{u_{i,j+1} - u_{i,j}}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j})^2 + (D_{+y}u_{i,j})^2}}\right. \\
&\quad \left. - \frac{u_{i,j} - u_{i,j-1}}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j-1})^2 + (D_{-y}u_{i,j})^2}}\right]
\end{aligned}
$$

Total variation denoising-ROF model

Then we use iteration method for the equation and get the following linearized equation:

$$
\begin{aligned}
u_{i,j}^{n+1} \quad = \quad & f_{i,j} + \frac{1}{2\lambda h^2}\Big[\frac{u_{i+1,j}^{n} - u_{i,j}^{n+1}}{\sqrt{\varepsilon^2 + (D_{+x}u_{i,j}^{n})^2 + (D_{0y}u_{i,j}^{n})^2}} \\
& - \frac{u_{i,j}^{n+1} - u_{i-1,j}^{n}}{\sqrt{\varepsilon^2 + (D_{-x}u_{i,j}^{n})^2 + (D_{0y}u_{i-1,j}^{n})^2}}\Big] \\
& + \frac{1}{2\lambda h^2}\Big[\frac{u_{i,j+1}^{n} - u_{i,j}^{n+1}}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j}^{n})^2 + (D_{+y}u_{i,j}^{n})^2}} \\
& - \frac{u_{i,j}^{n+1} - u_{i,j-1}^{n}}{\sqrt{\varepsilon^2 + (D_{0x}u_{i,j-1}^{n})^2 + (D_{-y}u_{i,j}^{n})^2}}\Big]
\end{aligned}
$$

Total variation denoising-ROF model

- c_1, c_2, c_3, c_4 are the coefficients.

$$u_{i,j}^{n+1} = \frac{2\lambda h f_{i,j} + c_1 u_{i+1,j}^n + c_2 u_{i-1,j}^n + c_3 u_{i,j+1}^n + c_4 u_{i,j-1}^n}{2\lambda h + c_1 + c_2 + c_3 + c_4}$$

Total variation denoising-ROF model

- About the h for an image of pixel-size $(M \times M)$. Some authors use $h = 1$, and the others use $h = 1/M$. If the the domain of the image f is always $\Omega = [0, 2^b] \times [0, 2^b]$, for an image of pixel-size $(M \times M)$ we should take $h = \frac{2^b}{M}$.
- About the λ.

Total variation denoising-ROF model

- The boundary condition is: for $1 \leqslant i, j \leqslant M-1$, let $u^n_{0,j} = u^n_{1,j}, u^n_{M,j} = u^n_{M-1,j}, u^n_{i,0} = u^n_{i,1}, u^n_{i,M} = u^n_{i,M-1}, u^n_{0,0} = u^n_{1,1}, u^n_{0,M} = u^n_{1,M-1}, u^n_{M,0} = u^n_{M-1,1}, u^n_{M,M} = u^n_{M-1,M-1}$.

```
function u=ROF(u0,IterMax,eps,lambda)
u0=double(u0);
[M N]=size(u0); %initialize u by u0 (not necessarily) or by a c
u=u0;
[M,N]=size(u);
h=1.;%  space discretization
for Iter=1:IterMax,
    Iter
    for i=2:M-1,
      for j=2:N-1,
    %----------computation of coefficients co1,co2,co3,co4---
    ux=(u(i+1,j)-u(i,j))/h;
    uy=(u(i,j+1)-u(i,j-1))/2*h;
        Gradu=sqrt(eps*eps+ux*ux+uy*uy);
        co1=1./Gradu;

        ux=(u(i,j)-u(i-1,j))/h;
    uy=(u(i-1,j+1)-u(i-1,j-1))/2*h;
        Gradu=sqrt(eps*eps+ux*ux+uy*uy);
        co2=1./Gradu;
```

```
ux=(u(i+1,j)-u(i-1,j))/2*h;
     uy=(u(i,j+1)-u(i,j))/h;
          Gradu=sqrt(eps*eps+ux*ux+uy*uy);
          co3=1./Gradu;

     ux=(u(i+1,j-1)-u(i-1,j-1))/2*h;
          uy=(u(i,j)-u(i,j-1))/h;
          Gradu=sqrt(eps*eps+ux*ux+uy*uy);
          co4=1./Gradu;

          co=1.+(1/(2*lambda*h*h))*(co1+co2+co3+co4);
     div=co1*u(i+1,j)+co2*u(i-1,j)+co3*u(i,j+1)+co4*u(i,j-1);
     u(i,j)=(1./co)*(u0(i,j)+(1/(2*lambda*h*h))*div);
       end
     end
```

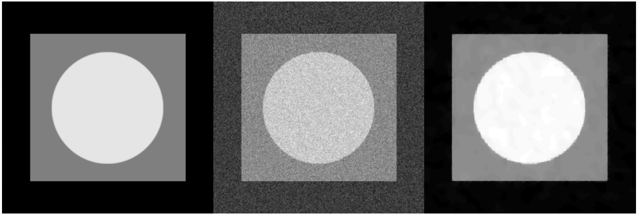
```
%------------ FREE BOUNDARY CONDITIONS IN u -------------------
    for i=2:M-1,
          u(i,1)=u(i,2);
          u(i,N)=u(i,N-1);
        end

    for j=2:N-1,
          u(1,j)=u(2,j);
          u(M,j)=u(M-1,j);
        end

        u(1,1)=u(2,2);
        u(1,N)=u(2,N-1);
        u(M,1)=u(M-1,2);
        u(M,N)=u(M-1,N-1);
```

```
%%% Compute the discrete energy at each iteration
en=0.0;
    for i=2:M-1,
      for j=2:N-1,
      ux=(u(i+1,j)-u(i,j))/h;
      uy=(u(i,j+1)-u(i,j))/h;
      fid=(u0(i,j)-u(i,j))*(u0(i,j)-u(i,j));
      en=en+sqrt(eps*eps+ux*ux+uy*uy)+lambda*fid;
      end
    end
%%% END computation of energy
Energy(Iter)=en;
end
% Plot the Energy versus iterations
figure
plot(Energy);legend('Energy/Iterations');
```

Figure : Gaussian:$\sigma = 0.02$ image



Contents

Chambolle's Projection Algorithm

- When $\phi(t) = t$ and R is the identity operator, Chambolle has remarked that the minimization of the total variation can be viewed as a projection problem on a suitable convex set.
- Chambolle and Lions proved that the following unconstrained minimization problem is referred to as the ROF model:

$$\min_{u \in BV(\Omega)} \int_{\Omega} |\nabla u(x)| \mathrm{d}x + \frac{\lambda}{2} \parallel u - f \parallel_2^2$$

for an adequate Lagrange multiplier $\lambda > 0$.

Chambolle's Projection Algorithm

- We will denote by $u_{i,j}$, $i, j = 1, \ldots, N$, a discrete image and by $X = \mathbb{R}^{N^2}$ the set of all discrete images of size N^2.
- $J(u) = \int_\Omega |\nabla u(x)| \mathrm{d}x$. Here the functional J is a discretization of the standard total variation.

$$J(u) = \sum_{1 \leqslant i,j \leqslant N} |(\nabla u)_{i,j}|$$

- The problem we want to solve is

$$\min_{u \in X} \{ J(u) + \frac{1}{2\lambda} \parallel u - f \parallel^2 \}$$

The unique minimizer of it is given by $u = f - P_{\lambda G}(f)$, where $P_{\lambda G}(f)$ is the L^2-orthogonal projection of f on the set λG.

Chambolle's Projection Algorithm

- Computing the nonlinear projection $P_{\lambda G}(f)$ amounts to solving the following problem:

$$\min\{|\lambda \mathrm{div} p - f|^2_{X\times X}; p \in X \times X, |P_{i,j}| \leqslant 1 \ \forall i,j = 1,\dots,N\}$$

For each i, j

$$-(\nabla(\lambda \mathrm{div}\, p - f))_{i,j} + \alpha_{i,j} p_{i,j} = 0$$

$$\alpha_{i,j} = |(\nabla(\lambda \mathrm{div}\, p - f))_{i,j}|$$

Chambolle's Projection Algorithm

Let $\tau > 0$ be given and let $p^0 = 0$ be an initial guess. We compute $p_{i,j}^{n+1}$ as

$$p_{i,j}^{n+1} = p_{i,j}^{n} + \tau((\nabla(\mathrm{div}p^n - f/\lambda))_{i,j} - |(\nabla(\mathrm{div}p^n - f/\lambda))_{i,j}|p_{i,j}^{n+1})$$

The final algorithm is described

$$p_{i,j}^{n+1} = \frac{p_{i,j}^{n} + \tau((\nabla(\mathrm{div}p^n - f/\lambda)))_{i,j}}{1 + \tau|(\nabla(\mathrm{div}p^n - f/\lambda)))_{i,j}|}$$

THEOREM Let us assume that $0 < \tau \leqslant \frac{1}{8}$. Then $\lambda \mathrm{div}p^n$ converges to $P_{\lambda G}(f)$ as $n \to \infty$.

Chambolle's Projection Algorithm

- The gradient $\nabla : X \to X \times X$

$$(\nabla u)^1_{i,j} = \begin{cases} u(i+1,j) - u(i,j) & \text{if } i < N \\ 0 & \text{if } i = N \end{cases}$$

$$(\nabla u)^2_{i,j} = \begin{cases} u(i,j+1) - u(i,j) & \text{if } j < N \\ 0 & \text{if } j = N \end{cases}$$

Chambolle's Projection Algorithm

- The divergence operator defined by analogy with the continuous case by div=$-\nabla^*$, where ∇^* is the adjoint of ∇.

$$(\operatorname{div} p)_{i,j} = (\operatorname{div} p)^1_{i,j} + (\operatorname{div} p)^2_{i,j}$$

$$(\operatorname{div} p)^1_{i,j} = \begin{cases} p(i,j)^1 - p(i-1,j)^1 & \text{if } 1 < i < N \\ p(i,j)^1 & \text{if } i = 1 \\ -p(i-1,j)^1 & \text{if } i = N \end{cases}$$

$$(\operatorname{div} p)^2_{i,j} = \begin{cases} p(i,j)^2 - p(i,j-1)^2 & \text{if } 1 < j < N \\ p(i,j)^2 & \text{if } j = 1 \\ -p(i,j-1)^2 & \text{if } j = N \end{cases}$$

```
function u=proj(f,t,lbd)
m=length(f);
p01=zeros(m,m);
p02=zeros(m,m);
n=1;
while n⩽100
    q0=div(p01,p02);
    u0=q0-f/lbd;
    [ux,uy]=grad(u0);
    V=(ux.^2+uy.^2).^(1/2);
    p11=(p01+t*ux)./(1+t*V);
    p12=(p02+t*uy)./(1+t*V);
    p01=p11;
    p02=p12;
    n=n+1;
end
p=div(p11,p12);
u=f-lbd*p;
```

```
function q=div(p1,p2)
n=length(p1);
q=zeros(size(p1));
for i=2:n-1
    for j=2:n-1
        q(i,j)=p1(i,j)-p1(i-1,j)+p2(i,j)-p2(i,j-1);
        q(1,j)=p1(1,j)+p2(1,j)-p2(1,j-1);
        q(n,j)=-p1(n-1,j)+p2(n,j)-p2(n,j-1);
    end
    q(i,1)=p1(i,1)-p1(i-1,1)+p2(i,1);
    q(i,n)=p1(i,n)-p1(i-1,n)-p2(i,n-1);
end
q(1,1)=p1(1,1)+p2(1,1);
q(1,n)=p1(1,n)-p2(1,n-1);
q(n,1)=-p1(n-1,1)+p2(n,1);
q(n,n)=-p1(n-1,n)-p2(n,n-1);
```

```
function [ux,uy]=grad(u)
n=length(u);
ux=zeros(size(u));
uy=zeros(size(u));
for i=1:n-1
    for j=1:n
        ux(i,j)=u(i+1,j)-u(i,j);
        ux(n,j)=0;
    end
end
for i=1:n
     for j=1:n-1
      uy(i,j)=u(i,j+1)-u(i,j);
      uy(i,n)=0;
     end
end
```