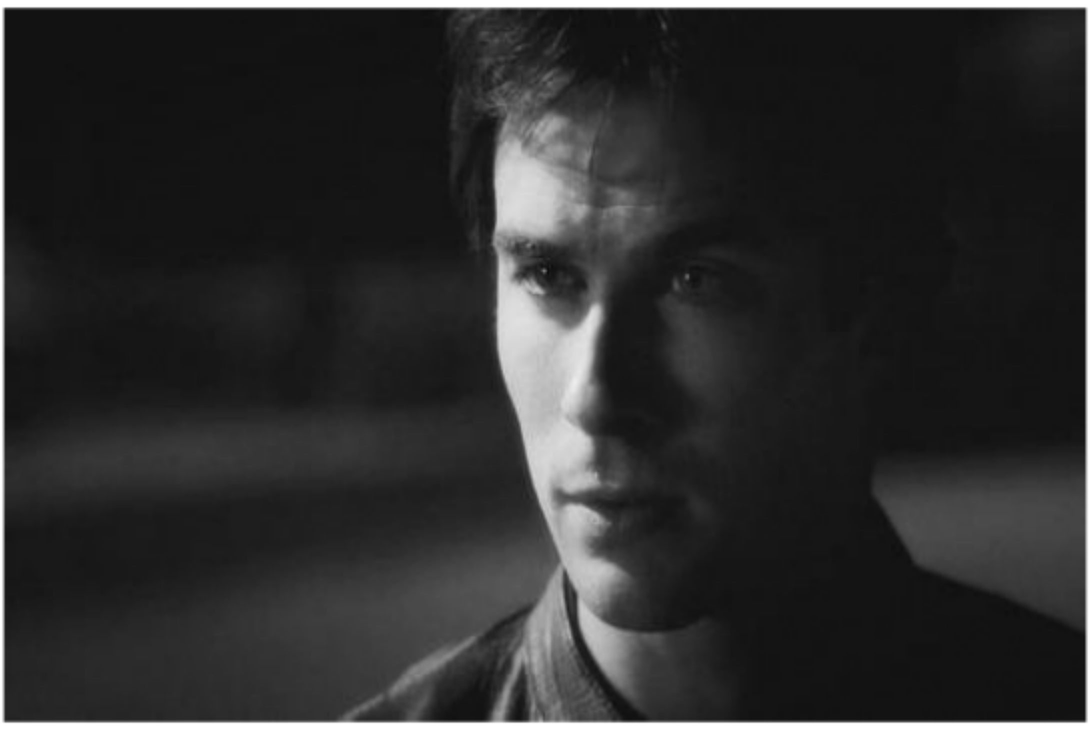
Accelerated Proximal Gradient for Image Restoration

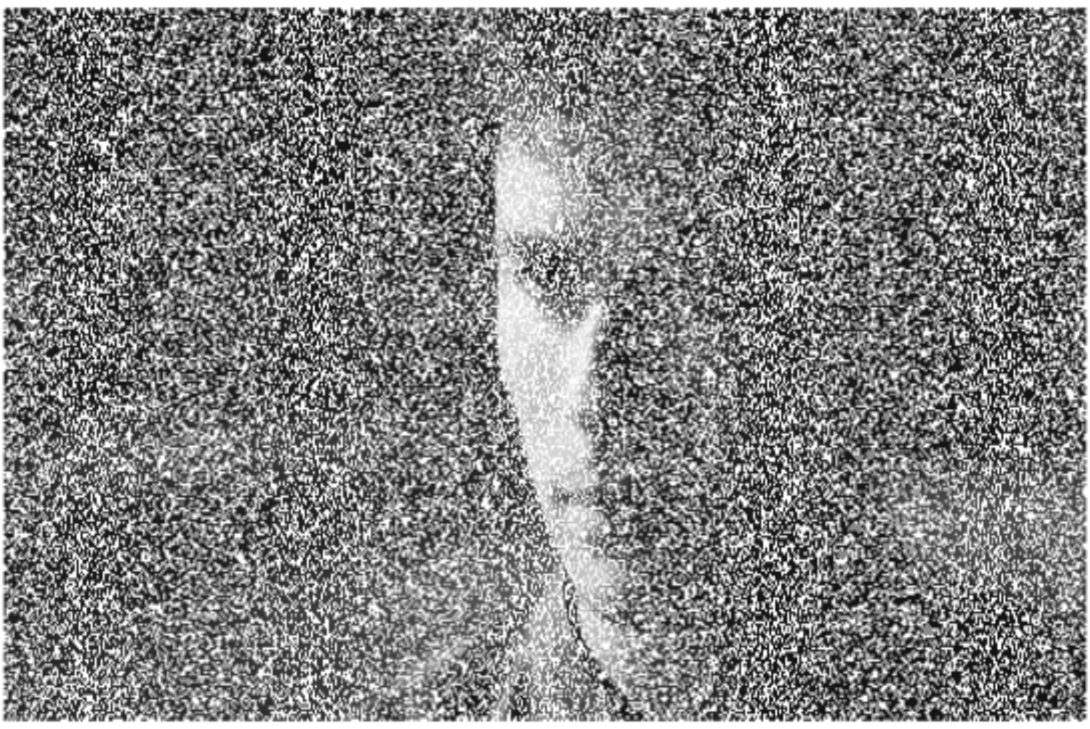
Huijuan Zhou

Nov.7,2017

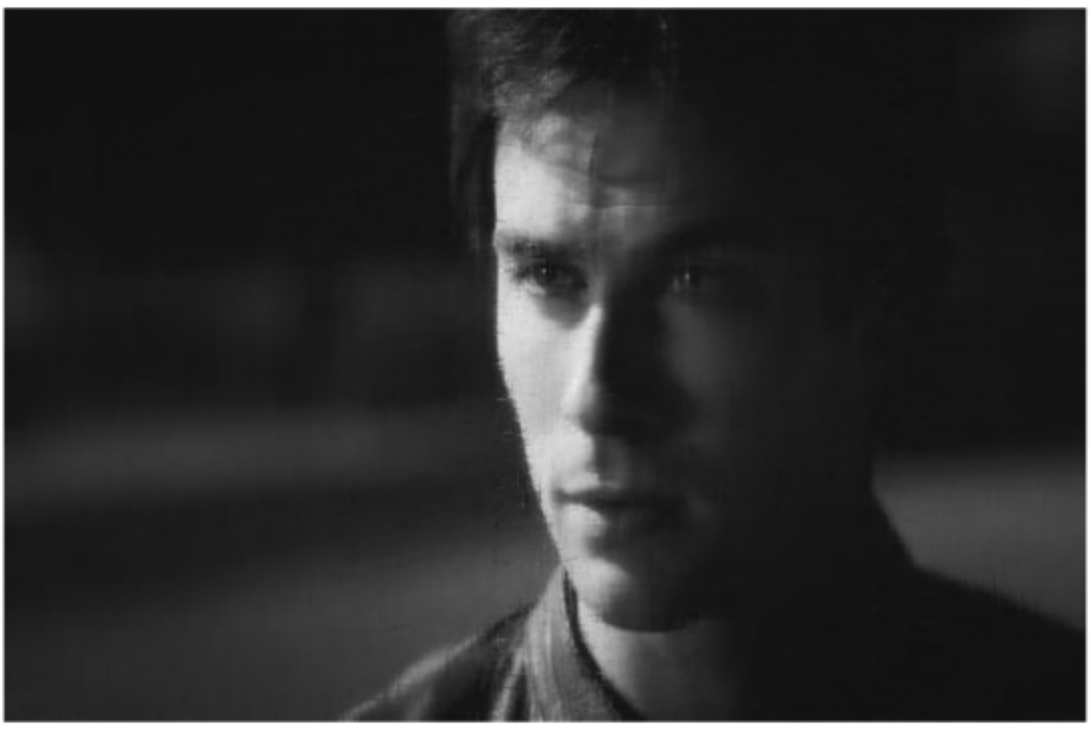
Original image: Lan Somerhalder



Damaged image



Reconstructed image



Theory

- For martrix $A \in R^{m \times n}$, the condensed SVD is $A = U_r \Sigma_r V_r$. For the constant $\lambda > 0$, the optimal solution of the optimization problem

$$\min_{X \in R^{m \times n}} f(X) = \frac{1}{2} ||X - A||_F^2 + \lambda ||X||_{S_1}$$

  is

$$\hat{X} = \mathcal{D}_\lambda(A) := U_r(\Sigma_r - \lambda I_r)_+ V_r^T,$$

where$(\Sigma_r - \lambda I_r)_+ = \text{diag}((\sigma_1 - \lambda)_+, \cdots, (\sigma_r - \lambda)_+)$.

$\hat{X}$ is the **singular value thresholding operator** of X.

- The elements of $M \in R^{m \times n}$ is partially missing, and

  M_{ij} is obversed if and only is $(i, j) \in \Omega$

The **optimization problem for image reconstruction** is

$$\min_{X \in R^{m \times n}} f(X) = \frac{1}{2} ||P_\Omega(X - M)||_F^2 + \lambda ||X||_{S_1}$$

Define $P_\Omega(Y) = \tilde{Y} = (\tilde{y}_{ij}) \in R^{m \times n}$, and

$$\tilde{y}_{ij} = \begin{cases} y_{ij}, & (i,j) \in \Omega \\ 0, & (i,j) \notin \Omega \end{cases}$$

- Let $f(X) = g(X) + h(X)$, where $g(X) = \frac{1}{2} ||P_\Omega(X - M)||_F^2$.

$$\nabla g(X) = P_\Omega(X - M), \nabla^2 g(X) \preceq K$$

where K can equal 1. So $g(X)$ can be controlled by

$$\tilde{g}(X) = g(Y) + < \nabla g(Y), X - Y > + \frac{K}{2} ||X - Y||_F^2$$

which equals

$$\tilde{g}(X) = C + \frac{K}{2} ||X - \{Y - \frac{1}{K} P_\Omega(Y - M)\}||_F^2$$

Replace $g(X)$ with $\tilde{g}(X)$, we get the **optimization problem**

$$\min_{X \in R^{m \times n}} \frac{1}{2} ||X - \{Y - \frac{1}{K} P_\Omega(Y - M)\}||_F^2 + \frac{\lambda}{K} ||X||_{S_1}$$

for each iteration.

Algorithm

- Initialize $Y_1 = M, X_1 = M, X_0 \in R^{m \times n}$, and $Y_0 \in R^{m \times n}$, where $M \in R^{m \times n}$ is partially missing and known;
- Loop:

  $X_0 = X_1$ (store the last result)

  $X_1 = \mathcal{D}_{\frac{\lambda}{K}}(Y_1)$ (proximal operator)

  $Y_0 = X_1 + \beta(X_1 - X_0)$ (Accelerate)

  $Y_1 = Y_0 + \frac{1}{K} P_\Omega(M - Y_0)$
- Until

  $||X_1 - X_0||_F < \epsilon$

Acceleration

parameter setting: $\lambda = 1, K = 1$, and

- $\beta = 0$

```
> proc.time()-time0
   用户    系统    流逝
 11.328   0.024  43.445
```

- $\beta = 0.2$

```
> proc.time()-time0
 用户   系统   流逝
9.660  0.004 18.606
```

- $\beta = 0.5$

```
> proc.time()-time0
 用户   系统   流逝
7.484  0.000 14.651
```

- $\beta = 0.8$

```
> proc.time()-time0
 用户   系统   流逝
9.956  0.000 17.449
```