

Accelerated Edge-Preserving Image Restoration Without Boundary Artifacts

Restauration d'images

Victor CILLEROS

Jérémy JEAN

Maxime GEY

Présentation du problème

- ❖ Image convoluée avec un flou gaussien
- ❖ Résolution avec des conditions circulaires au bord de l'image : peu réaliste
- ❖ L'approche proposée résout le problème dans un cas plus général



Fomulation et Résolution

Formulation

❖ On cherche à minimiser :

$$x = \operatorname{argmin}_x \{ \psi(x) = J(x) + \lambda \phi(Rx) \}$$

↑ Terme de régularisation

$$J(x) = \left\| y - TAx \right\|_2^2$$


Image floutée Opérateur de masque Image reconstruite matrice de flou circulante

C'est un problème d'optimisation non-trivial -> plusieurs stratégies de résolution

Résolution

❖ Approches existante : SALSA/SPLIT-BELLMAN

- Séparation du terme de régularisation avec une variable auxiliaire

SALSA $u = x$  $\min_{x,u} \{ \psi(x, u) = \frac{1}{2} \| y - TAx \|_2^2 + \lambda \phi(Ru) \}$

SPLIT-BELLMAN $v = Rx$  $\min_{x,v} \{ \psi(x, v) = \frac{1}{2} \| y - TAx \|_2^2 + \lambda \phi(v) \}$

- Expression du Lagrangien associé (AL : Augmented Lagrangian)

$$L(x, u, \mu, \eta) = \psi(x, u) + \frac{\mu}{2} \| u - x - \eta \|_2^2 \qquad L(x, v, \mu, \eta) = \psi(x, v) + \frac{\mu}{2} \| v - Rx - \eta \|_2^2$$

Résolution

❖ Approche proposée : ADMM-P2

- Séparation du terme de régularisation avec deux variable auxiliaire

$$\min_{x, u_0, u_1} \{ \psi(u_0, u_1) = \frac{1}{2} \| y - Tu_0 \|_2^2 + \lambda \phi(u_1) \} \quad \text{avec} \quad \begin{array}{l} u_0 = Ax \\ u_1 = Rx \end{array}$$

- Forme compacte

$$\min_{x, w} \psi(w) \quad \text{avec} \quad \begin{array}{l} w = Dx \\ D = [A, R]^T \\ x = [u_0, u_1]^T \end{array}$$

Résolution

❖ **Approche proposée : ADMM-P2**

- Expression du Lagrangien

$$L(x, w, \gamma) = \psi(w) + \frac{\mu}{2} \left\| w - Dx - \eta \right\|^2$$

Implémentation

Algorithme

❖ ADMM-P2

Select $x^0, \nu > 0, \mu > 0$

Precompute $T'y$

Set $\eta_0^{(0)} = 0, \eta_1^{(0)} = 0, k = 0$

Repeat

$$u_0^{k+1} = (T'T + \mu I_N)^{-1} [T'y + \mu(Ax^{(k)} + \eta_0^{(k)})]$$

$$u_1^{(k+1)} = \text{shrink}\{Rx^{(k)} + \eta_1^{(k)}, \frac{\lambda}{\mu}\}$$

$$x^{(k+1)} = H_\nu^{-1} [A'(u_0^{(k+1)} - \eta_0^{(k)} + \nu R'(u_1^{(k+1)} - \eta_1^{(k)}))]$$

$$\eta_0^{(k+1)} = \eta_0^{(k)} - (u_0^{(k+1)} - Ax^{(k+1)})$$

$$\eta_1^{(k+1)} = \eta_1^{(k)} - (u_1^{(k+1)} - Rx^{(k+1)})$$

$$k = k + 1$$

Avec

$$H_\nu = A'A + \nu R'R$$

Implémentation

❖ Propriétés de matrice circulaire

Pour A une matrice circulante : $A = F^{-1}DF$

Avec D matrice diagonale

Opérateur h de convolution $h = \begin{pmatrix} a & b & a \\ b & c & b \\ a & b & a \end{pmatrix}$

On utilise par la suite : $Ax = h \otimes x$

$= F^{-1}(F(h) \cdot F(x))$ Produit terme à terme

$= F^{-1}(\text{diag}(F(h)) \cdot F(x))$ Pré-compute

Implémentation

- ❖ Pour chaque produit de matrice :

```
def fft_dot(self, x:np.ndarray, F:np.ndarray)->np.ndarray:  
    return np.fft.ifft2(np.multiply(F, np.fft.fft2(x))).real
```

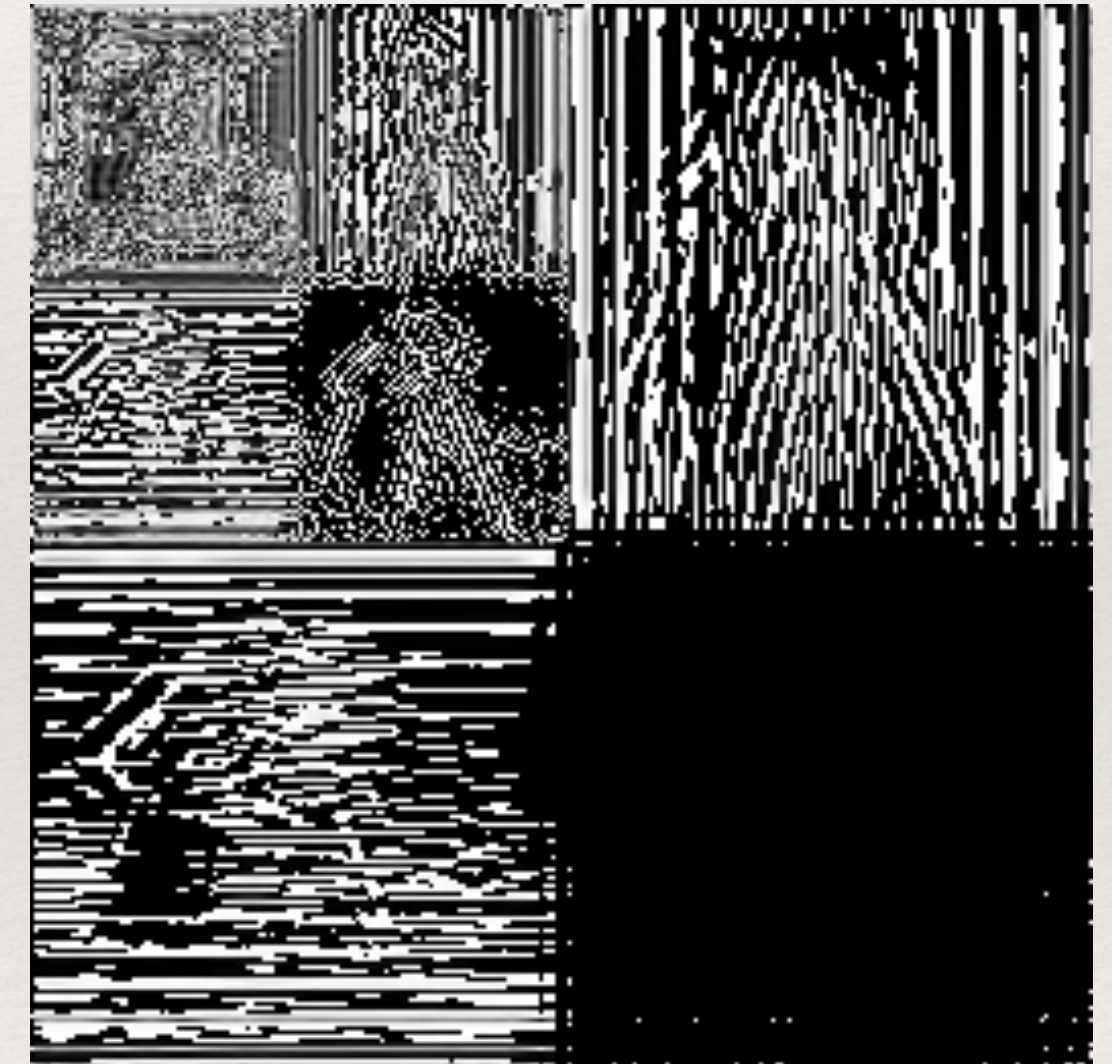
```
def rfft_dot_adj(self, x:np.ndarray, F:np.ndarray)->np.ndarray:  
    return np.fft.irfft2(np.multiply(np.conj(F), np.fft.rfft2(x)))
```


Implémentation

- ❖ Régularisation : utilisation de « wavelet »
Package pywt pour la fonction :



pywt.wavedec2(x, wavelet='db2', level=2, mode='zero')



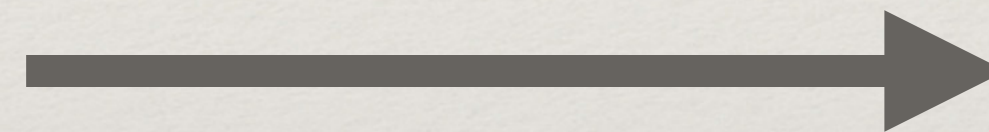
Résultats

Paramètres de départ

- ❖ Choix des paramètres : en théorie convergence garantie indépendamment du choix

$$\nu_{min} = \operatorname{argmin}_{\mu} k(A'A + \nu R'R)$$

$$\nu\mu = 2^8 \frac{\lambda \nu_{min}}{x_{max}}$$



$$\mu_0 = 2^{-4}$$

$$\nu = 2^8 \frac{\lambda \nu_{min}}{\mu_0 x_{max}}$$

Evolution de l'erreur



Evolution du BSNR



Commentaires

- ❖ Possibilité d'utiliser la TV pour la régularisation (plus adaptée à notre image?)
- ❖