Analysis of Motion Blur Using Double Discrete Wavelet Transform

Yi Zhang
University of Dayton, stander@udayton.edu

Follow this and additional works at: http://ecommons.udayton.edu/stander_posters

Recommended Citation
http://ecommons.udayton.edu/stander_posters/190
Object motion causes spatially varying blur. Estimating such a type of blur from a single image is an ill-posed problem that is difficult to solve. We introduce the notion of double discrete wavelet transform (DDWT) designed to sparsely represent the blurred image and blur kernel simultaneously. Based on DDWT analysis, we are able to accurately estimate motion blur kernels and recover the latent sharp image. The blind image deblurring solution we propose handles spatially varying motion blur effectively and efficiently.

### 1. Common Blur Types
- **Cause**: Pixel records light from multiple sources.
- **Three** common types of blur:
  - Camera shake
  - Ghosting
  - Object motion

### 2. Motion Model
- **Sharp image**: \(x[n]\).
- **Blur kernel**: \(b[n] = k^{-1}(\text{degradation} - \text{blur} + k)\).
- **Observed image**: \(a[n] = (\mathcal{F} * b)[n] + z[n]\), where \(z[n]\) is noise.

### 3. Double Discrete Wavelet Transform
**Definition 1**: Translation-invariant discrete wavelet transform (DWT) is defined by the relation:
\[
\tilde{a}[j] = \mathcal{F}^{i}[b[n]],
\]
where \(i, j\) are subband indices, and \(\mathcal{F}\) is the wavelet analysis filter.

**Definition 2**: Double discrete wavelet transform (DDWT) is defined by the relation:
\[
\tilde{v}[i][j] = \mathcal{F}^{2}[a[n]],
\]
where \(i, j\) are subband indices, and \(\mathcal{F}\) is the wavelet analysis filter.

### 4. DDWT Analysis of Blurry Image
**DDWT and its equivalent process**:

![Diagram of DDWT analysis and equivalent process](image)

**Idea**: detect double edges ↔ detect motion blur kernel.

**Input**: 
- Zoom-in DWT \(a[n]\)
- Zoom-in DDWT \(\tilde{a}[n]\)
- Left-Right Matting \(\beta[n]\)

### 5. Autocorrelation Analysis
- **No blur**:
  \[
  \hat{R}_v(0, \ell) = R_v(0, \ell) + R_v(\ell, \ell) - R_v(\ell, 0) - \hat{R}_v(\ell, \ell). 
  \]

- **Blur**:
  \[
  \hat{R}_v(0, \ell) = \frac{1}{2} \left( 2R_v(0, \ell) - R_v(\ell, \ell) - \hat{R}_v(\ell, 0) \right) = R_v(0, \ell). 
  \]

**Second local minimum** ↔ motion blur length:
\[
\ell = \arg \min_{\ell \in \{0, \ldots, L\}} \hat{R}_v(0, \ell). 
\]

### 6. Local Analysis
**Idea**: averaging window over “same object” ↔ local window \(\Lambda_v\) contains pixels similar to \(v[n]\).

\[
R_v(\ell, \sigma, n) = \frac{1}{|\Lambda_v|} \sum_{n' \in \Lambda_v} v[n'] + m[n] \sigma + \ell 
\]

### 7. Left-Right Matting
- **Goal**: recover sharp DWT coefficients from DDWT coefficients.
- **Idea**: classify each DDWT coefficient as the left or right.

\[
\beta[n]\]

- **Input**: 
  - Left-side estimates:
    \[
    \tilde{v}[n]_{\text{left}} = \frac{\sum_{n' \in \Lambda_v} \tilde{v}[n']}{|\Lambda_v|} 
    \]
  - Right-side estimates:
    \[
    \tilde{v}[n]_{\text{right}} = \frac{\sum_{n' \in \Lambda_v} \tilde{v}[n']}{|\Lambda_v|} 
    \]

### 8. Image Deblurring
1. **Denoise**:
   \[
   \hat{v}[n] = \text{wavelet}_{\text{deblur}}(v[n]) 
   \]

2. **Left-right estimates**:
   \[
   \hat{u}[n]_{\text{left}} = \frac{\sum_{n' \in \Lambda_v} \hat{u}[n']}{|\Lambda_v|}, \quad \hat{u}[n]_{\text{right}} = \frac{\sum_{n' \in \Lambda_v} \hat{u}[n']}{|\Lambda_v|} 
   \]

3. **Reconstruction of \(u[n]\)**:
   \[
   \tilde{u}[n] = \frac{\hat{u}[n]_{\text{left}} + \hat{u}[n]_{\text{right}}}{2} 
   \]