Fast Direct Super-Resolution by Simple Functions
Chih-Yuan Yang and Ming-Hsuan Yang
Electrical Engineering and Computer Science, University of California, Merced
(cyang35, mhyang)@ucmerced.edu
code available http://eng.ucmerced.edu/people/cyang35

Challenges
• How to generate super-resolution images quickly?
• How to upsample low-resolution images effectively?
• How to work for various images?

Main Ideas
• Use simple features and direct mapping function to reduce computational load
• Segment low-resolution feature space into numerous subspaces and training functions individually for effective mapping
• Exploit a large set of training images to learn robust mapping functions

Results of BSD 200 Images
The first 20 test images

Algorithm
(1) Extract features: intensity minus mean. Only a central region of high-resolution patches is used for better accuracy
(2) Learn cluster centers in the LR feature space
(3) Learn linear regression coefficients
\[ C^* = \arg\min_C \left\| W - C \begin{pmatrix} V \end{pmatrix} \right\|^2 \]
\[ W: \text{features of } n \text{ high-resolution patches} \]
\[ V: \text{features of } n \text{ low-resolution patches} \]
1: a vector with all values as 1
Number of training patches
\[ w = C^* \begin{pmatrix} 1 \end{pmatrix} \]
(4) For every low-resolution test patch, find the closet cluster center. Reconstruct high-resolution features using the learned coefficients
(5) Compute output intensities as the mean of overlapping patches

Experimental Results (factor=4)

Conclusions
• Simple, fast, and effective super-resolution algorithm
• Easy to implement

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PSNR</th>
<th>SSIM index</th>
<th>Language</th>
<th>Exe. Time*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic Interp</td>
<td>24.27</td>
<td>0.6555</td>
<td>MATLAB</td>
<td>0.005</td>
</tr>
<tr>
<td>Irani91</td>
<td>25.01</td>
<td>0.7036</td>
<td>MATLAB</td>
<td>1.52</td>
</tr>
<tr>
<td>Shan08</td>
<td>23.47</td>
<td>0.6367</td>
<td>C</td>
<td>37.50</td>
</tr>
<tr>
<td>Sun08</td>
<td>24.54</td>
<td>0.6695</td>
<td>MATLAB</td>
<td>77.82</td>
</tr>
<tr>
<td>Kim08</td>
<td>25.12</td>
<td>0.6970</td>
<td>MATLAB+C</td>
<td>26.95</td>
</tr>
<tr>
<td>Glasner09</td>
<td>25.20</td>
<td>0.7064</td>
<td>MATLAB</td>
<td>1586</td>
</tr>
<tr>
<td>Yang10</td>
<td>24.31</td>
<td>0.6205</td>
<td>MATLAB</td>
<td>31.57</td>
</tr>
<tr>
<td>Friedman11</td>
<td>22.22</td>
<td>0.6173</td>
<td>MATLAB</td>
<td>1454</td>
</tr>
<tr>
<td>Wang12</td>
<td>24.32</td>
<td>0.6505</td>
<td>MATLAB+C</td>
<td>1695</td>
</tr>
<tr>
<td>Proposed</td>
<td>25.18</td>
<td>0.7081</td>
<td>MATLAB</td>
<td>10.02</td>
</tr>
</tbody>
</table>

*in second for a factor of 4 on a 2.7GHz Quad Core machine.