Classification of Line and Character Pixels on Raster Maps Using Discrete Cosine Transformation Coefficients and Support Vector Machines
The Problem

• To understand the information on raster maps
  – How? Recognize the line and characters on the raster map for further processing
The Problem

• To understand the information on raster maps
  – How? Recognize the line and characters on the raster map for further processing
The Problem

• To understand the information on raster maps
  – How? Recognize the line and characters on the raster map for further processing
Related Work

• Steps to recognize the lines and characters:
  – FIND AREAS of characters
  – For each area, SEPARATE and REBUILD lines and characters
  – Send characters to Optical Character Recognition component
  – Send lines to Vectorization component

• These steps are interrelated
Related Work

• Some of the work assume that the line and character pixels are not overlapping (Bixler00, Fletcher88, Velazquez03)

• Li et al. work in local areas to separate the characters from lines

• Cao et al. use the different length of line segments to separate characters from line arts
Related Work

• They all based on geometric properties
  – The size of a character
  – The size of a word (string)
  – The size of the gap between characters
  – The size of the gap between words
  – etc.

• They assume the foreground can be easily separated from the background
Our Approach

• We use texture classification approach to classify pixels on the raster maps
Our Approach

• Features:
  – Discrete Cosine Transformation (DCT) coefficients

• Classifier:
  – Support vector machine
Discrete Cosine Transformation

• DCT – Discrete Cosine Transformation
  – DCT is closely related to the discrete Fourier transform (DFT)
  – The discrete cosine transform (DCT) is a technique for converting a signal into elementary frequency components
Discrete Cosine Transformation

- DCT gives us the strength of each component to build a single image

\[
S(u, v) = \frac{2}{\sqrt{nm}} C(u)C(v) \sum_{u=0}^{K} \sum_{v=0}^{K} s(x, y) \cos \frac{(2x + 1)u\pi}{2n} \cos \frac{(2y + 1)v\pi}{2m}, \\
\quad \quad \quad \quad u = 0, K, n - 1; \quad v = 0, K, m - 1
\]
Discrete Cosine Transformation
Remove background

- We apply DCT transformation for each pixel
- The DCT coefficients represent the variation around each pixel
- The pixels with low variation (near 0) around them are the background pixels
Remove background

- Now we have the color of the background pixels by DCT
- The probability of color C to be background $P(B|C)$ and the probability of the color to be foreground $P(F|C)$
  - If $P(B|C) > P(F|C)$ then color C is background color
  - Else color C is foreground color
Remove background
Classify Line and Character pixels

- We apply DCT transformation for each foreground pixel
- The DCT coefficients represent the variation around each foreground pixel
- We use the DCT coefficients as features for SVM to classify the pixels
Classify Line and Character pixels

- Training
  - One MapQuest map for character samples
  - One Google map and one Viamichline map for line samples
Classify Line and Character pixels

• Training
  – One MapQuest map for character samples
  – One Google map and one Viamichline map for line samples
Classify Line and Character pixels

- Training
  - One MapQuest map for character samples
  - One Google map and one Viamichline map for line samples
Classify Line and Character pixels

• Classification
  – The testing maps are disjoint from the training samples
Classify Line and Character pixels

• Classification
  – The testing maps are disjoint from the training samples
Classify Line and Character pixels

• Classification
  – The testing maps are disjoint from the training samples
Classify Line and Character pixels

<table>
<thead>
<tr>
<th>Map Source</th>
<th>Precision/Recall of Classification</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Line Pixels</td>
<td>Character Pixels</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ours</td>
<td>Cao's</td>
<td>Ours</td>
<td>Cao's</td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>99/91%</td>
<td>95/91%</td>
<td>79/98%</td>
<td>77/85%</td>
<td></td>
</tr>
<tr>
<td>MSN</td>
<td>99/79%</td>
<td>91/87%</td>
<td>75/99%</td>
<td>81/86%</td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>99/99%</td>
<td>95/99%</td>
<td>98/99%</td>
<td>95/72%</td>
<td></td>
</tr>
<tr>
<td>Yahoo</td>
<td>95/91%</td>
<td>70/96%</td>
<td>91/92%</td>
<td>88/30%</td>
<td></td>
</tr>
<tr>
<td>Mapquest</td>
<td>99/78%</td>
<td>88/73%</td>
<td>84/98%</td>
<td>76/85%</td>
<td></td>
</tr>
<tr>
<td>Map24</td>
<td>95/74%</td>
<td>97/70%</td>
<td>73/96%</td>
<td>70/98%</td>
<td></td>
</tr>
<tr>
<td>ViaMichelin</td>
<td>83/34%</td>
<td>44/57%</td>
<td>87/96%</td>
<td>90/68%</td>
<td></td>
</tr>
<tr>
<td>Multimap</td>
<td>89/82%</td>
<td>98/64%</td>
<td>63/90%</td>
<td>46/97%</td>
<td></td>
</tr>
<tr>
<td>TIGER/Line</td>
<td>99/94%</td>
<td>97/89%</td>
<td>83/99%</td>
<td>67/90%</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>98/85%</td>
<td>85/82%</td>
<td>83/96%</td>
<td>71/71%</td>
<td></td>
</tr>
</tbody>
</table>
Discussion

• Computation time:
  – For a 400x400 Google Map:
    • 2 seconds to remove background
    • 4 seconds to classify line and character pixels

• No threshold needed

• Line and character pixels can be used in vectorization and OCR components